УДК 004.451.44

ИЕРАРХИЧЕСКОЕ ПЛАНИРОВАНИЕ ЗАДАЧ В СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ НА МНОГОЯДЕРНЫХ ПРОЦЕССОРАХ

© 2016 С.В. Осмоловский, Е.Р. Иванова, И.Р. Федоров

Санкт-Петербургский государственный университет аэрокосмического приборостроения

Статья поступила в редакцию 28.03.2016

В статье рассматривается ряд аспектов, связанных с планированием задач на многоядерных процессорах, существующие типы многопроцессорных систем и виды многопроцессорного планирования. Основное внимание будет сконцентрировано на описании, анализе и сравнении наиболее популярных схем иерархического планирования на многоядерных процессорах.

Ключевые слова: *иерархическое планирование, системы реального времени, многоядерные процессоры, временное разделение*

В наши дни микропроцессорная отрасль бурно развивается, благодаря чему многопроцессорные (multiprocessor) и многоядерные (multicore) платформы становятся привлекательным решением для удовлетворения растущих требований по производительности вычислительных систем. К многоядерным архитектурам проявляется значительный интерес, так как проблемы тепловыделения и энергопотребления ограничивают дальнейшее увеличениепроизводительности одноядерных процессоров. Тенденция к переходу встраиваемых систем (embedded systems) к использованию многоядерных аппаратных платформ позволяет повысить производительность такой системы, а также реализовать больше функциональных возможностей в рамках единой универсальной вычислительной системы [1]. В настоящее время встраиваемые системы становятся неотъемлемой частью многих современных комплексов, в том числе и систем реального времени (СРВ, real-time systems), обладающих повышенными требованиями к безопасности (safety-critical). В связи с этим, наблюдается возрастающий интерес к теории планирования задач реального времени на многоядерных платформах и практической реализации полученных методов и подходов [2].

В отраслях промышленности, где используются системы для ответственного применения, такие, как авионика, космическая отрасль и автомобилестроение, безопасность совместной работы нескольких программных логических блоков в пределах одной системы может быть достигнута за счет использования иерархического планирования (hierarchical scheduling) прикладных задач (task), из которых состоят программные приложения. В таких системах разделение ресурсов (процессорное время, память, каналы обмена) между логическими блоками обеспечивается использованием планирования с временным разделением (time partitioning) на системном уровне [3]. Метод иерархического планирования является одним из главных способов создания приложений с четко разделенными программными логическими блоками - разделами (partition) [4]. Разделы имеют выделенные ресурсы (область памяти, процессорное время в течение каждого цикла системы, доступ к каналам обмена и пр., что обеспечивает временное разделение, благодаря которому приложения, выполняющиеся в одном разделе, не будут нарушать корректность использования любого разделяемого ресурса с приложениями других разделов. Приложения, структурированные таким образом, являются более надежными, чем одноуровневые («плоские», flat), т.к. благодаря иерархическому планированию ошибки и сбои не будут распространяться за пределы конкретного раздела. Это также позволяет производить независимую от других разделов верификацию и валидацию приложений (что упрощает процесс сертификации).

Согласно методу иерархического планирования глобальный планировщик (планировщик разделов) распределяет вычислительные ресурсы между разделами в зависимости от их периода и времени выполнения. На втором уровне локальный планировщик (планировщик задач) запускает задачи внутри раздела в течение временного окна, предоставленному текущему разделу глобальным планировщиком. Иерархическое планирование используется для обеспечения корректной работы в одном приложении задач жесткого (hard), мягкого (soft) и нереального времени [5]. Иерархическое планирование также используется в системах смешанной критичности (mixed-criticality), т.е. системах, где одновременно могут функционировать задачи с различными уровнями критичности по безопасности (mixed-safety), защищенности (mixed-security), отказоустойчивости и ряду других параметров. Изоляция, которую предоставляет иерархическое планирование, также облегчает повторное использование программного обеспечения.

Свойства и архитектурные особенности метода иерархического планирования отвечают большому количеству требований и предписаний аэрокосмических стандартов и спецификаций (ARINC-653[6], AIR [7], MILS и т.д.), а также методу пространственного и временного разделения (time and space partitioning, TSP) [8], использующегося в концепции Интегральной Модульной Авионики (ИМА, Integrated Modular Avionics, IMA) [9], которая успешно применяется в авиации и внедряется в космические программы [10]. Сутью ИМА является переход к универсальным вычислительным системам, обладающим возможностью одновременного исполнения задач различного рода и уровня критичности. А это, в свою очередь, позволяет соблюдать такие важные требования, выдвигаемые современным встраиваемым системам, как возможность гибкого распределения ресурсов, уменьшения размера, веса и потребляемой мощности (Size, Weight & Power, SWaP). Следовательно, иерархическое планирование

Осмоловский Сергей Валерьевич, младший научный сотрудник. E-mail: sergey.osmolovskiy@guap.ru
Иванова Екатерина Романовна, инженер. E-mail: ekaterina.ivanova@guap.ru
Федоров Иван Романович, инженер. E-mail: ivan.fedorov@guap.ru

может быть успешно использовано в современных встраиваемых системах с повышенными требованиями к безопасности.

Цель работы: описать метод иерархического планирования, области его применения, провезти анализ и сравнение наиболее популярных схем иерархического планирования задач на многоядерных процессорах.

Планирование задач реального времени на многоядерных процессорах.

A. Планирование реального времени. В СРВ временная предсказуемость (predictability), выраженная в форме гарантий того, что каждая задача будет иметь требуемое время отклика (response time), которое уложится в строгие критические сроки (дедлайн, deadline), столь же важна, как производительность (то, как быстро каждая задача будет завершена) или пропускная способность (количество задач, которое может быть завершено за некий заданный период времени). Переход от выполнения одной задачи к другой осуществляется в результате планирования (sche-duling). В научно-исследовательских работах, посвящённых СРВ, большое внимание уделяется алгоритмам планирования – набору правил, согласно которому производится переключение задач и выделение им требуемых ресурсов. Эти алгоритмы базируются на модели задач реального времени, которые содержат важную информацию о временных характеристиках и параметрах задач в СРВ [11]. Алгоритмы планирования должны предоставлять априорные доказательства того, что все задачи будут успевать завершать свое выполнение относительно дедлайнов, при условии, что все задачи удовлетворяют свойствам и характеристикам, определенным в модели задач. Эти гарантии должны быть аналитически доказаны до начала фактического функционирования СРВ (в режиме офлайн).

Можно выделить два типа алгоритмов планирования: политики планирования (sche-duling policies) и тесты планируемости (sche-dulability tests). Политики планирования, которые иногда называют планировщиками (schedu-lers), контролируют расписание задач. Они работают в системе одновременно с задачами и в режиме онлайн принимают решения по планированию, основываясь на характеристиках и ограничениях задач. Тесты планируемости до запуска системы производят проверку соблюдения гарантий выполнения задач к заданным критическим срокам. С точки зрения теории планирования реального времени планирование называется выполнимым (feasible), если все задачи могут быть завершены в соответствии с набором указанных ограничений. Набор задач называют планируемым (schedulable), если существует хотя бы один алгоритм, который может производить выполнимое планирование.

B. Модель задач реального времени. В большом количестве тематических научно-исследовательских работ СРВ описываются набором независимых периодических/спорадических задач, которые должны быть планируемы на одном или нескольких процессорах в соответствии с их временными параметрами [12]. Экземпляр задачи называется работой (job). Задача может порождать бесконечное количество работ. Задача имеет следующие временные параметры:

- Дедлайн D_i: Критический срок, к которому задача должна успеть выполниться.
- Период T_i: Минимальный временной интервал между активизациями экземпляров задачи - моментами порождения работ (release time, arrival time).
- Наихудшее время исполнения задачи (worst-case execution time) C_i: Наихудшее значение времени исполнения задачи без ее приостановок и прерываний.

Простым способом проведения проверки планируемости n периодических задач с неявным дедлайном ($T_i = D_i$) на процессоре является расчет коэффициента использования процессора (processor utilization factor) U согласно формуле (1).

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \qquad (1)$$

Если $U > 1$, то задачи не могут быть успешно планируемы на данном процессоре.

C. Виды многопроцессорных систем. В многопроцессорных вычислительных платформах (многоядерных процессорах) для выполнения задач доступно несколько процессоров (процессорных ядер). В теории планирования СРВ на многоядерных процессорах различают по крайней мере три вида многопроцессорности [13]:

- Идентичная (identical): процессоры полностью идентичны, обладают одинаковыми функциональными возможностями и частотой.
- Однородная (uniform): процессоры обладают одинаковыми функциональными возможностями, но разной частотой.
- Неоднородная (unrelated): процессоры имеют разные функциональные возможности и частоту. Задачи могут быть не в состоянии выполняться на всех процессорах.

Виды многопроцессорности приведены на рис. 1.

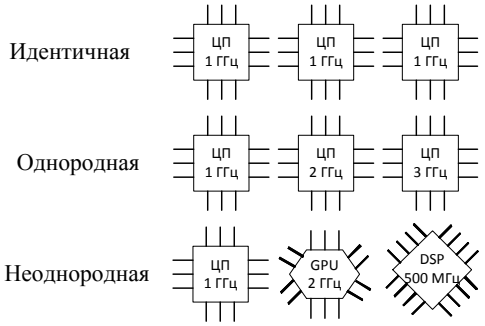

Рис. 1. Виды многопроцессорности

D. Типы многопроцессорного планирования. Существующие на данный момент способы планирования реального времени на многоядерных процессорах можно разделить на три категории: распределенное (partitioned, non-migrating), совместное (global, full migrating) и гибридное (hybrid, restricted migrating) планирование.

1. Распределенное многопроцессорное планирование. При распределенном многопроцессорном планировании набор задач размещается в нескольких непересекающихся множествах и для каждого множества назначается свой процессор (процессорное ядро). На каждом процессоре присутствует планировщик с собственной очередью задач, готовых к исполнению, и задачи / работы не могут перемещаться («мигрировать») на другие процессоры. Таким образом, когда распределение задач между процессорами выполнено, данный тип многопроцессорного планирования позволяет применять существующие алгоритмы планирования и тесты планируемости реального времени, созданные для однопроцессорных систем. Это является

весомым преимуществом данного способа планирования. В данном типе планирования отсутствуют накладные расходы, связанные с миграцией задач между процессорами. Еще одним преимуществом распределённого планирования является отсутствие в нем так называемого «эффекта Дала» – аномалии планирования, сформулированной Dhall и Liu [14]. Тем не менее, распределенное планирование по сравнению с совместным, имеет важный недостаток: расходование процессорных ресурсов далеко не всегда происходит эффективно. Может возникнуть ситуация, когда внутри многопроцессорной системы некоторые процессоры простаивают без выполнения полезной работы – исполнения задач, а некоторые, наоборот, не могут исполнять задачу, так как для нее не хватает вычислительных мощностей данного процессора. А миграция задач с одного процессора на другой для балансировки нагрузки недопустима. Также при распределении задач между ядрами необходимо решать NP-трудную проблему упаковки в контейнеры (bin-packing) в режиме офлайн.

2. Совместное многопроцессорное планирование. При совместном многопроцессорном планировании задачи планируются, находясь в единой очереди готовности к исполнению, и могут перемещаться между процессорами. Основным преимуществом совместного планирования является наличие возможности преодоления алгоритмической сложности, присущей распределенному планированию. Поскольку все процессоры используют единую очередь задач, готовых к исполнению, это избавляет от необходимости решения проблемы распределения задач между процессорами в офлайн режиме, которая является источником вычислительной сложности распределенного планирования. Другим важным преимуществом совместного планирования является то, что, как правило, при его использовании требуется производить меньше операций вытеснения (preemption) задач: планировщик будет вытеснять задачу только когда нет простаивающих (не выполняющих полезную работу, idle) процессоров. Но в отличие от распределенного планирования, алгоритмы однопроцессорного планирования не подходят для использования при совместном планировании из-за «эффекта Дала». Также при совместном планировании необходимо решать задачу корректной работы с разделяемой памятью (shared-memory).

3. Гибридное многопроцессорное планирование. При совместном планировании накладные расходы мигрирующих задач могут быть очень высоки (зависит от архитектуры многопроцессорной системы). По факту, задержки связаны с промахами кэша (cache miss) и коммуникационными нагрузками. А это, в свою очередь, может повлечь увеличение наихудшего времени исполнения задачи, что является крайне нежелательным в СРВ. С другой стороны, распределенные алгоритмы часто страдают от потерь вычислительной мощности, когда нагрузка на процессоры распределяется неравномерно. Чтобы преодолеть все эти недостатки предлагается использовать гибридные подходы планирования: частично распределенный (semi-partitioned) и кластерный (clustering).

Частично распределенное планирование. При частично распределенном планировании большинство задач выполняются только на одном процессоре, как и при классическом распределенном планировании. Однако несколько задач (или работ) могут мигрировать между двумя и более процессорами. Основная идея подобного метода планирования заключается в повышении коэффициента использования благодаря применению совместного планирования для тех задач, которые не могут быть закреплены только за одним процессором из-за ограничений эвристических алгоритмов решения задачи об упаковке в контейнеры, что является важным преимуществом. Задачи, которые не могут быть полностью закреплены за одним процессором, будут расщепляться (split) и связываться с несколькими процессорами. Но процесс связывания задач с процессорами происходит в режиме офлайн, что можно отнести к недостаткам данного подхода. На данный момент исследования в области повышения эффективности разбиения задач продолжаются, позволяя сокращать накладные расходы, связанные с миграцией и вытеснением задач.

Кластерное планирование. Кластерное планирование также с**оч**етает в себе преимущества распределенного и совместного планирования. Основной идеей кластерного планирования является разделение m процессоров на $\left\lceil \frac{m}{c} \right\rceil$ наборов по с процессоров в каждом, образуя кластеры [15]. И распределенный, и совместный методы планирования можно рассматривать как частные случаи кластерного при $c = 1$ и $c = m$ соответственно. Изначально кластерное планирование было схоже с распределенным планированием, где задачи связывались с процессорами в режиме офлайн. В случае с кластеризацией задачи связывались с конкретным кластером, которому затем выделялся набор процессоров. Такой подход упрощал решение задачи упаковки в контейнеры для распределенного метода, и теперь задачи должны были быть распределены по кластерам. Для распределения задач между кластерами, повышения коэффициента использования, снижения накладных расходов, связанных с миграцией задач и уменьшением их времени отклика, могут применяться различные эвристические алгоритмы. Каждый кластер обрабатывает небольшое количество задач на небольшом количестве выделенных процессоров и тем самым устраняет проблему длинной очереди задач на исполнение, с которыми часто сталкиваются алгоритмы совместного планирования. Кластеризация также имеет определенную гибкость с точки зрения возможности создания кластеров для различных типов задач и задач с большим или маленьким значением коэффициента использования. Другим преимуществом кластеризации является то, что можно создать кластеры с различным объемом ресурсов, например, кластеры с большим или маленьким количеством процессоров, имеющих один и тот же кэш второго уровня и т.д.

Иерархическое планирование.

А. Описание. Иерархическое планирование является весьма популярным направлением научных исследований в области СРВ. Основной идеей данного подхода является разделение ресурсов на четко определенные разделы (контейнеры, подсистемы). Этот метод активно используется в аэрокосмической промышленности для обеспечения изоляции и предотвращения распространения появляющихся ошибок в другие части системы, а также для упрощения анализа и сертификации системы [16]. Иерархическое планирование представляет собой удобный механизм для обеспечения модульности программных приложений в СРВ, гарантируя временное разделение между приложениями. Такой подход был введен для обеспечения возможности совместного использования процессора приложениями реального времени, требующих различных политик планирования. Одним из главных

преимуществ иерархического планирования является то, что оно обеспечивает средство для разбиения сложной системы на разделы. В соответствии с этим подходом, система может быть иерархически разделена на ряд разделов, которые планируются с помощью глобального планировщика. Каждый раздел содержит набор задач, которые планируются локальным планировщиком. Иерархическое планирование позволяет разделам разрабатываться и анализироваться по отдельности со своим собственным локальным планировщиком, а затем, используя глобальный планировщик, оно обеспечивает объединение нескольких разделов в единую систему, не нарушая временных характеристик данных разделов, которые были проанализированы ранее по отдельности. Объединение предполагает и проведение теста планируемости всей системы (на глобальном уровне), проверяя удовлетворение всех временных требований.

Таким образом, иерархическое планирование представляет собой систему, состоящую из крупномодульных разделов с предсказуемым временным поведением. Иерархическое планирование имеет ряд преимуществ помимо улучшения времени отклика задач. Оно предоставляет возможность параллельной разработки и тестирования разделов, упрощает объединение разделов (анализ планируемости), поддерживает временное разделение во время функционирования системы и безопасное выполнение задач, может облегчить изолирование возникающих ошибок и сбоев, упрощает проведение структурного анализа и повторного использования уже разработанных приложений [17]. Иерархическое планирование также облегчает повторное использование разделов, поскольку их вычислительные требования характеризуются четко определенными интерфейсами. Реализации иерархического планирования называются схемами иерархического планирования (СИП, Hierarchical Scheduling Framework, HSF).

В. Стратегии иерархического планирования. Для реализации иерархического планирования могут быть использованы следующие стратегии [18]:

- планирование на основе серверов;
- композиционное планирование.

1. Иерархическое планирование на основе серверов. Алгоритмы на базе сервера изначально использовались для обслуживания апериодических задач. Развитие данных алгоритмов, а также более глубокое изучение их изоляционных свойств позволяет успешно применять серверы (например, такие типы, как сервер пропускной способности (bandwidth server), спорадический сервер (sporadic server) и т.д.) при иерархическом планировании. При этом они становятся составными частями протоколов резервирования ресурсов (resource reservation protocols). Каждый сервер связан с бюджетом (budget, capacity) процессорного времени и периодом. Бюджет времени – это время работы сервера. Если задача не успевает выполниться в пределах, то (в зависимости от типа сервера) происходит заем времени у следующего периода сервера. Бюджет времени (опять же, в зависимости от типа сервера) пополняется не в каждом периоде сервера, а в определенное время пополнения (replenishment time) на рассчитываемую величину пополнения (replenishment amount) [19].

Согласно иерархическому планированию на основе серверов существует один или несколько глобальных планировщиков на первом уровне и несколько серверов на втором уровне. Каждый сервер связан с очередью готовых к исполнению на сервере задач и с определенной политикой планирования. Глобальный планировщик пополняет бюджет каждого сервера, т.е. выделяет ему процессорное время, устанавливает для них дедлайны и диспетчирует все готовые серверы согласно своей политике планирования. Локальные планировщики организуют планирование задач, которые закреплены за конкретным сервером. На всех уровнях алгоритмы планирования могут быть произвольными. СИП, использующая серверы, изображена на рис. 2.

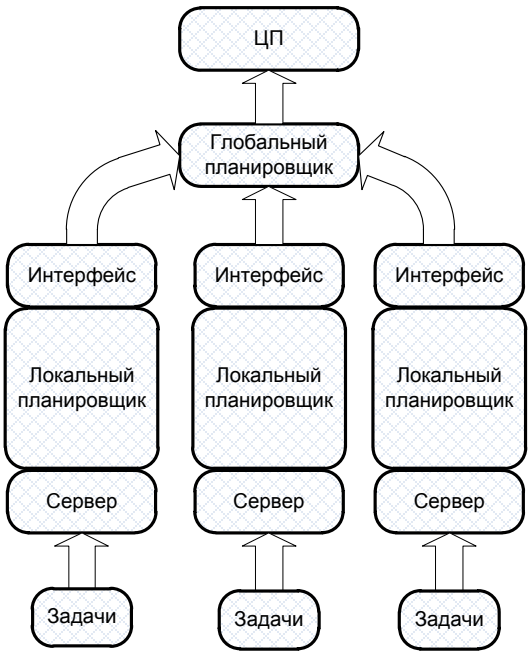

Рис. 2. Схема иерархического планирования, использующая серверы

Сервер считается готовым к исполнению, если его очередь готовности не пуста. Задачи на готовом к планированию сервере исполняются до момента завершения бюджета сервера или до появления более приоритетного сервера. Интерфейс сервера определяет количество процессоров, которые будут зарезервированы для конкретного сервера. Большинство типов серверов обладает быстрым временем отклика, а также обеспечивает предсказуемость и устойчивость поведения СРВ в случаях, когда превышена максимальная скорость поступления задач. Основным недостатком данной стратегии является сложность управления и анализа смешанных моделей задач.

2. Композиционное иерархическое планирование. Основной идеей композиционного иерархического планирования является объединение классического и широко используемого принципа «разделяй и властвуй» с временными требованиями. Эта стратегия была принята в качестве методологии при проектировании больших и сложных систем на основе систематической абстракции и композиции. Согласно данной стратегии сложность каждого компонента системы скрыта и абстрагирована через упорядоченный и хорошо определенный интерфейс. К преимуществам данной стратегии следует отнести четкое разграничение нюансов планирования между разработчиками разделов и системным интегратором. Разработчики разделов не должны предоставлять сведения о внутреннем содержимом разделов (характеристиках и

атрибутах задач и т.д.), должен быть просто абстрактный интерфейс раздела. С другой стороны, можно выделить следующие недостатки данной стратегии:

- Чем больше в системе разделов, тем меньше обеспечивается загрузка процессора.
- Использование совместных ресурсов между разделами может быть сложным в реализации и проведении анализа планируемости.
- Некоторые выполнимые системы не могут быть планируемы с этой стратегией.
- Ограничения, наложенные на периоды разделов, необходимы для составления эффективного расписания. В противном случае, полученное расписание может быть далеко не оптимальным.

IV. Иерархическое планирование на многоядерных процессорах.

A. Обзор. В настоящее время существует много вариантов реализации СИП. Мы рассмотрели и провели сравнительный анализ наиболее известных СИП для СРВ на многоядерных процессорах, предложенных Shin и др. [20], Åsberg и др.[21], Checconi и др.[22] и Nemati и др. [23].

1. Иерархическое планирование с виртуальными кластерами. Схема иерархического планирования с виртуальными кластерами (СИП-ВК, Virtual Cluster Hierarchical Scheduling Framework), разработанная Shin, является обобщением подхода физической кластеризации с добавлением возможности совместного использования процессоров между различными кластерами. В отличие от физических кластеров, где процессоры для них выделяются в режиме офлайн, данная схема позволяет распределять процессоры между кластерами в процессе работы. Для организации подобной динамической схемы распределения необходим интерфейс для обеспечения требований по выполнению и параллелизму в пределах кластера, использующий иерархическое планирование. Этот интерфейс был предложен Shin и известен как Многопроцессорный Периодический Ресурс (МПР, Multiprocessor Periodic Resource).

В СИП-ВК для каждого кластера создается МПР-интерфейс, используя анализ планируемости, предложенный Shin. Затем каждый интерфейс превращается в набор периодических серверов с неявным дедлайном для внутрикластерного планирования. Периодические серверы «простаивают», если активная задача, выполняющаяся внутри сервера, отсутствует. СИП-ВК имеет двухуровневую иерархию планировщиков: межкластерный (inter-cluster, глобальный) и внутрикластерный (intra-cluster, локальный) планировщики. Межкластерный уровень организовывает динамическое распределение виртуальных кластеров, т.е. производит операцию связывания физического процессора с виртуальным кластером. Каждому разделу (группе задач и серверу), планируемому согласно расширению классического алгоритма планирования Earliest Deadline First (EDF) [24] до совместного типа многопроцессорного планирования - Global EDF (G-EDF) [25], назначается подмножество процессоров. То же самое верно и для межкластерного планировщика в рассматриваемой схеме, за исключением того, что каждый кластер может содержать до *m* активных серверов. Все серверы от всех кластеров помещены в очередь в соответствии с глобальной политикой планирования. Очередь задач сортируется по наиболее раннему дедлайну и задачи распределяются по процессорам до тех пор, пока все процессоры не будут заняты. Отметим, что задачи, не связанные с каким-либо конкретным сервером, принадлежат определенному кластеру. Внутрикластерный уровень обеспечивает планирование на физическом уровне в кластере. Кластеры не пересекаются и, следовательно, не могут прерывать друг друга. На этом уровне тоже используется алгоритм планирования G-EDF. Внутрикластерный планировщик исполняет задачи, закрепленные за кластером, используя бюджет планирующих их серверов. В отличие от обычного иерархического планирования, внутрикластерный планировщик также должен использовать алгоритм планирования для многопроцессорных систем, так как у кластера может быть несколько активных серверов. Архитектура СИП-ВК изображена на рис. 3. В результате СИП-ВК может быть описана как двухуровневое совместное планирование. Easwaran и др. изучали различные алгоритмы совместного планирования, такие как G-EDF и алгоритм McNaughton's [26], которые можно было бы использовать в СИП-ВК.

2. Схема иерархического планирования Checconi. *Checconi* предложил собственный вариант СИП, состоящей из двух уровней иерархических планировщиков: локального и глобального. На локальном уровне задачи планируются с помощью совместного многоядерного алгоритма со статическими приоритетами. На глобальном уровне каждый процессор имеет собственный Hard Constant Bandwidth Server (H-CBS) планировщик, который диспетчирует по одному серверу в каждом разделе. Каждый раздел имеет число серверов равное числу процессоров и обладает доступом ко всем процессорам. Совместное планирование запускается параллельно с H-CBS планировщиками. H-CBS жестко связан с процессором. Задачи в пределах одного сервера могут перемещаться на другой сервер, который находится на другом процессоре. Сервер имеет в своем распоряжении часть каждого процессора.

Рис. 3. Архитектура СИП-ВК

3. Схема иерархического планирования Nemati. Другая СИП была предложена *Nemati*. В его схеме серверы планируются согласно совместному многопроцессорному типу планирования (со статическими и динамическим приоритетами), а на локальном уровне каждый раздел планируется согласно распределенному многопроцессорному типу планирования (со статическими и динамическим приоритетами), т.е. каждый раздел имеет максимум один сервер (он может исполняться на любом процессоре), задачи всегда исполняются на одном и том же процессоре.

4. Последовательная схема иерархического планирования. Еще одна СИП - это Последовательная СИП (П-СИП, Sequential HSF), предложенная Asberg и

др. Этот подход обеспечивает планирование серверов последовательно, таким образом, планирование задач в каждом разделе происходит с использованием совместных алгоритмов планирования на всех процессорах. Кроме того, данная схема будет выполнять разделы последовательно, занимая каждый процессор одним сервером.

В. Сравнение. Все рассмотренные выше СИП были реализованы для идентичных многопроцессорных систем. Сложность реализации этих СИП различается как на локальном, так и на глобальном уровнях. СИП-ВК имеет наиболее сложный планировщик серверов (на глобальном уровне), так как может быть задействовано наибольшее количество процессоров, занятые процессоры должны быть проверены при планировании сервера. СИП Checconi является более простой, так как все серверы связываются с процессором статически в режиме офлайн. Действительно, одного глобального планировщика H-CBS может быть достаточно для обработки всех серверов, что делает его похожим на П-СИП. Подход Nemati довольно прост, поскольку существует максимум один сервер на разделы, однако необходимо постоянно проверять доступность процессоров, так как серверы назначаются на них не статически, а динамически. П-СИП проста, так как серверы назначаются на процессоры статически.

Таблица 1. СИП для многоядерных процессоров

Схема	Многопроцессорное планирование	Параллелизм серверов	Назначение серверов процессорам
СИП-ВК	кластерное	есть	онлайн
СИП Checconi	совместное	есть	офлайн
СИП Nemati	раздельное	есть	онлайн
П-СИП	совместное	нет	офлайн

Взглянув на локальное планирование, можно сказать, что СИП-ВК и СИП Checconi имеют схожую схему планирования на локальном уровне. Обе используют совместное многопроцессорное планирование на подмножестве процессоров. СИП Nemati использует обычное распределенное планирование, в то время как П-СИП – совместное многопроцессорное планирование на всех ядрах. Подводя итог, в СИП-ВК и СИП Checconi планирование наиболее алгоритмически сложное, чем в П-СИП, а СИП Nemati является наиболее простой. С точки зрения общих (разделяемых) ресурсов, так как СИП Nemati использует распределенное планирование, то работа с разделяемыми ресурсами внутри раздела становится менее сложной в данном подходе по сравнению с тремя другими схемами. Краткий обзор результатов сравнения четырех СИП для многоядерных процессоров приведен в табл. 1.

Выводы: рассмотрены основные аспекты, свойства и архитектурные нюансы метода иерархического планирования задач, обеспечивающего предсказуемое поведение системы и временное разделение между приложениями. Были представлены существующие виды многопроцессорности и типы многоядерного планирования. Было приведено описание и проведен сравнительный анализ существующих СИП (СИП-ВК, СИП Checconi, СИП Nemati, П-СИП) для идентичных многопроцессорных систем, выделены их особенности, преимущества и недостатки, которые могут быть важны для области их применения. Согласно приведенным характеристикам СИП, они могут применяться в качестве базовой программной архитектуры в СРВ для ответственных применений. Разработчики программного обеспечения для подобных систем смогут определить, какой тип планирования и СИП лучше использовать для решения конкретных задач, состава и характеристик целевой СРВ.

Вопросы, рассматриваемые в статье, тесно связаны с аспектом временного разделения, имеющего важное значение для обеспечения исполнения гарантий реального времени и для проведения независимого временного анализа приложения, требуемых во многих системах для ответственных применений, в том числе и аэрокосмической отрасли. Тем самым иерархическое планирование удовлетворяет значительному количеству требований аэрокосмических стандартов и спецификаций. Отметим, что спецификация ARINC-653 описывает двухуровневую СИП с использованием концепции разделов, а стандарт MILS требует модульности приложений и также оперирует понятием раздела.

Иерархическое планирование является актуальным направлением научно-исследовательских работ в области СРВ и обладает большим потенциалом практического использования в системах для ответственного применения во многих отраслях промышленности.

СПИСОК ЛИТЕРАТУРЫ:

1. *Boudjadar, A.* Schedulability and Energy Efficiency for Multi-core Hierarchical Scheduling Systems / *A. Boudjadar, A. David, J. Kim* et al. // Proceedings of the International Congress on Embedded Real Time Software and Systems ERTS, 2014. P. 35-44.
2. *Chakma, K.A.* Hierarchical Scheduling Approach for Symmetric Multiprocessing Based Real Time Systems on VxWorks / *K. Chakma, S. Debbarma, N. Kar* et al. // Lecture Notes on Software Engineering. 2013. Vol. 1, No. 1. P. 61-65.
3. *Leiner, B.* A comparison of partitioning operating systems for integrated systems. / *B. Leiner, M. Schlager, R. Obermaisser, B. Huber* // F. Saglietti and N. Oster, editors, SAFECOMP, volume 4680 of Lecture Notes in Computer Science. – Springer, 2007. P. 342-355.
4. *Ашневиц, Д.Н.* Архитектура операционной системы реального времени с изолированными разделами / *Д.Н. Ашневиц, С.В. Осмоловский* // Научная сессия ГУАП, 7-11 апреля 2014. – СПб.: ГУАП, 2014. С. 3-8.

5. *Kavi, K.* Real-Time Systems: An Introduction and the State-of-the-art / *K. Kavi, R. Akl.* – Wiley Encyclopedia of Computer Science and Engineering, 2008. P. 2369–2377.
6. ARINC 653: Avionics Application Software Standard Interface (Draft 15). Airlines Electronic Engineering Committee (AEEC), 1996. P. 14-1-14-7.
7. *Rufino, J.* Architecting robustness and timeliness in a new generation of aerospace systems / *J. Rufino, J. Craveiro, and P. Verissimo* // Architecting Dependable Systems VII, ser. Lecture Notes in Computer Science, A. Casimiro, R. de Lemos, and C. Gacek, Eds. – Springer, 2010. Vol. 6420. P. 146–170.
8. *Windsor, J.* Time and space partitioning in spacecraft avionics / *J. Windsor, K. Hjortnaes* // Proceedings of the 3rd IEEE International Conference on Space Mission Challenges for Information Technology, Pasadena, CA, USA, Jul. 2009. P. 13-20.
9. AEEC, "Design guidance for Integrated Modular Avionics," Aeronautical Radio, Inc., ARINC Report 651-1, Nov. 1997. P. 88.
10. *Rushby, J.* Partitioning in avionics architectures: Requirements, mechanisms and assurance // SRI Int'l., California, USA, NASA Contractor Report CR-1999-209347, 1999. P. 58
11. *Осмоловский, С.В.* Методы планирования задач в операционных системах реального времени // Научная сессия ГУАП: Ч.1 Технические науки. Сборник докладов 9-11 апреля 2012 г. – СПб.: ГУАП, 2012. С. 100-104.
12. *Li, H.* Scheduling mixed-criticality Real-Time Systems // Ph.D. dissertation, University of North Carolina at Chapel Hill, 2013. 104 p.
13. *Brandenburg, B.* Scheduling and Locking in Multiprocessor Real-Time Operating Systems // Ph.D. dissertation, University of North Carolina at Chapel Hill, 2011. 614 p.
14. *Dhall, S.* On a real-time scheduling problem / *S. Dhall, C.L. Liu* // Operations Research.1978. V. 26(1). P. 127–140.
15. *Calandrino, J.* A hybrid real-time scheduling approach for large-scale multicore platforms. / *J.M. Calandrino, J.H. Anderson, D.P. Baumberger* // Real-Time Systems, 2007. ECRTS '07. 19th Euromicro Conference on, April. P. 247-258.
16. *Asberg, M.* On the Development of Hierarchical Real-Time Systems // Licentiate Thesis, Mälardalen University, Sweden, 2012. P. 156.
17. *Asberg, M.* A Loadable Task Execution Recorder for Hierarchical Scheduling in Linux / *M. Asberg, T. Nolte, S. Kato* // Proc. of the Embedded and Real-Time Computing Systems and Applications (RTCSA). 2011. V. 1. P. 380-387.
18. *Crespo, A.* Mixed criticality in control systems / *A. Crespo, A. Alonso, M. Marcos* et al. // Proc. 19th World Congress The Federation of Automatic Control. 2014. P. 12261-12271.
19. *Осмоловский, С.В.* Метод многоуровневого планирования задач в операционных системах реального времени для многоядерных процессоров / С.В. Осмоловский // Научная сессия ГУАП: Ч.1 Технические науки. Сборник докладов 6-10 апреля 2015 г – СПб.: ГУАП, 2015. С. 109-116.
20. *Shin, I.* Hierarchical scheduling framework for virtual clustering of multiprocessors. / *I. Shin, A. Easwaran, I. Lee* // Proceedings of the 2008 Euromicro Conference on Real-Time Systems. IEEE Computer Society, 2008. P. 181-190.
21. *Asberg, M.* Towards Hierarchical Scheduling in Linux/Multi-core Platform / *M. Åsberg, T. Nolte, S. Kato*// MRTC Mälardalen University, Technische Universiteit Eindhoven, Sweden, The University of Tokyo, 2010. P. 1-4.
22. *Checconi, F.* Hierarchical Multiprocessor CPU Reservations for the Linux Kernel / *F. Checconi, T. Cucinotta, D. Faggioli, G. Lipari* // Proc. of the 5th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications, June 2009. P. 15-22.
23. *Nemati, F.* Multiprocessor Synchronization and Hierarchical Scheduling / *F. Nemati, M. Behnam, T. Nolte* // Proc. of the 1st International Workshop on Real-time Systems on Multicore Platforms: Theory and Practice, in conjunction with ICPP'09, September 2009. P. 58-64.
24. *Liu, C.* Scheduling Algorithms for Multi-Programming in a Hard-Real-Time Environment / *C. Liu, J. Layland* // ACM. 1973. Vol. 20, No. 1. P. 46–61.
25. *Peloquin, M.*A Comparison of Scheduling Algorithms for Multiprocessors // December 2010. P. 17.
26. *Abdullah, S.* Virtual Clustered-based Multiprocessor Scheduling in Linux Kernel / S. Abdullah // Ph.D. dissertation, Malardalen University, June 2013. 97 p.

HIERARCHICAL TASKS SCHEDULING IN REAL TIME SYSTEMS ON MULTICORE PROCESSORS

© 2016 S.V. Osmolovskiy, E.R. Ivanova, I.R. Fedorov
St. Petersburg State University of Aerospace Instrumentation

In article a number of the aspects connected with planning of tasks on multicore processors, the existing types of the multiprocessor systems and types of the multiprocessor scheduling is considered. The main attention will be concentrated on the description, analysis and comparison of the most popular schemes of hierarchical scheduling on multicore processors.

Key words: *hierarchical scheduling, real time systems, multicore processors, temporal partitioning*

Sergey Osmolovskiy, Minor Research Fellow.
E-mail: sergey.osmolovskiy@guap.ru
Ekaterina Ivanova, Engineer. E-mail: ekaterina.ivanova@guap.ru
Ivan Fedorov, Engineer. E-mail: ivan.fedorov@guap.ru