УДК 681.3.012
ББК 332

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ УСТРОЙСТВА АППАРАТНОЙ ПОДДЕРЖКИ ДИСПЕТЧЕРА ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ ПАКЕТА CPN TOOLS

*© **Н.П. Вашкевич**, Пензенский государственный университет (г. Пенза, Россия)*

*© **Р.А. Бикташев**, Пензенский государственный технологический университет (г. Пенза, Россия)*

*© **Е.А. Кизилов**, Пензенский государственный университет (г. Пенза, Россия)*

SIMULATION DEVICE HARDWARE SUPPORT TASK MANAGER USING THE PACKAGE CPN TOOLS

*© **N.P. Vashkevich**, Penza State Universiti (Penza, Russia)*

*© **R.A. Biktashev**, Penza State Technological Universiti (Penza, Russia)*

*© **E.A. Kyzilov**, Penza State Universiti (Penza, Russia)*

В статье предложена методика преобразования алгоритма задачи, представленного моделью недетерминированного автомата, в модель на основе сети Петри с реализацией и имитационным моделированием в программном пакете CPN TOOLS. На примере алгоритма диспетчеризации задач показано применение методики для получения динамических характеристик и оценки производительности разрабатываемого устройства аппаратной поддержки диспетчера с разделением времени для многопроцессорной системы. В процессе моделирования динамических характеристик получены данные о работоспособности устройства аппаратной поддержки диспетчера задач в составе управляющего и операционного блоков.

Ключевые слова: диспетчер задач, многопроцессорная система, имитационное моделирование, сеть Петри.

Technique transform algorithm of task that is represented by the model nondeterministic automata, Petri nets-based model and simulation with the CPN TOOLS software package. The example shows how to use the task scheduling algorithm method for dynamic characteristics and performance of the hardware device manager support for multiprocessor time-sharing system. In the process of modeling of dynamic characteristics of obtained data on the health of hardware support device Task Manager in managing and operating units.

Key words: Multiprocessor system, Task Manager, network simulation, Petri net.

Введение

В настоящее время практически в любой операционной системе на программном уровне реализован диспетчер, управляющий назначением задач процессору. В многопроцессорных системах диспетчер с разделением времени имеет единственную очередь готовых задач [1], которая является общим ресурсом, поэтому в алгоритм работы вводят средства синхронизации, разрешающие конфликты. В качестве таких средств обычно используют монитор, который представляется в виде процедуры, вызываемой программой диспетчеризации в точках синхронизации. Кроме того, при реализации алгоритма диспетчеризации между задачами и процессорами возникают отношения типа «клиент – сервер», которые также требуют синхронизации и которые зачастую выполняются с использованием метода «рандеву» [2].

Программная реализация диспетчера хорошо отработана в связи с широким распространением, но требует значительных временных затрат на выполнение процедур, связанных с вхождением процесса в монитор и реализацию очереди блокированных процессов, которая возникает из-за конкуренции множества процессоров при доступе к диспетчеру. Эти обстоятельства многократно увеличивают время ожидания процессов и, следовательно, снижают общую производительность многопроцессорной системы. Одним из путей решения проблемы является аппаратная поддержка функций диспетчеризации,

которая в значительной степени снимет проблему временных потерь. Кроме этого, применение аппаратной поддержки приводит к увеличению надежности операционной системы, что особенно важно для проблемно-ориентированных и специализированных систем. Предлагаемый подход в данной работе заключается в том, что аппаратная поддержка диспетчера задач выполняется в виде независимого устройства в составе многопроцессорной системы [3].

Сложная система синхронизации взаимодействующих процессов вызывает необходимость проверки правильности функционирования алгоритмов устройства диспетчеризации в динамическом режиме совместно с другими устройствами многопроцессорной системы. Для этого предлагается использовать метод имитационного моделирования, так как поведение системы в целом зависит как от корректной работы конечного автомата, реализующего устройство управления, так и от управляемого им операционного устройства. Поведенческая имитационная модель позволяет оценить динамические характеристики системы, выявить достаточность или избыточность сигналов управления, получить оценки показателей производительности, загрузки отдельных блоков и др.

Инструментарием имитационного моделирования сетей Петри является программная система CPN Tools. В ней применен аппарат функциональных, цветных, иерархических, временных сетей Петри [4], использование которых позволяет строить системы любой сложности, выявлять свойства алгоритмов, а при задании задержек на переходах будет моделироваться динамика функционирования системы.

В качестве примера рассмотрим моделирование устройства аппаратной поддержки диспетчера задач с разделением времени, используемого во многих операционных системах из-за наличия возможности автоматического поддержания балансировки нагрузки процессоров [1].

В общем случае алгоритм работы устройства следующий. При поступлении задачи определяется свободное место в очереди задач. Если места нет, то задача на обслуживание не принимается, иначе задача ставится в конец очереди. При наличии свободного процессора задача, стоящая в голове очереди, отправляется на обслуживание. Если свободны одновременно несколько процессоров, то производится выбор одного из них для обслуживания очередной задачи.

1. Формализация алгоритма синхронизации процессов при диспетчеризации задач в многопроцессорных системах

Формальное описание алгоритма взаимодействия процессов в данной задаче базируется на использовании моделей недетерминированных автоматов [НДА] [5]. Модели представляются в виде систем канонических уравнений, описывающих все реализуемые события управляющего алгоритма. Алгоритм работы диспетчера содержит две части: клиентскую и серверную. Клиентская часть связана с постановкой задачи в очередь и запросом процессора. Серверная часть связана с выбором процессора из пула свободных и собственно с самой процедурой обслуживания (обработки задачи).

При реализации алгоритма используются два метода синхронизации. Первый метод, основанный на взаимном исключении, обеспечивает корректный выбор процессора для обслуживания очередной задачи, второй – «рандеву» – определяет моменты одновременного наличия задачи в очереди и готовности одного из процессоров к обслуживанию этой задачи.

Для описания алгоритма введены основные частные события, представленные в табл. 1 [3]. В последней графе таблицы представлены сигналы, действующие в CPNTools-модели, имитирующие соответствующие им события модели НДА.

Таблица 1 – Таблица событий НДА и соответствующих им сигналов

События в НДА-модели	Описание событий	Сигналы в имитационной модели
S_Q^t	Задача в очереди	*t_S_Q*
S_{ZP}^t	Запрос процессора диспетчером	*t_S_ZP*

S_{GPj}^{t}	Задача готова к обслуживанию в *j*-м процессоре	*t_S_GPj*
S_{PZ}^{pj}	Подтверждение запроса *j*-м процессором	*pj_S_PZ*
S_{S}^{pj}	Свободен *j*-й процессор	*pj_S_S*
S_{S}^{p}	В пуле имеются свободные процессоры	*p_S_s*
S_{PT}^{pj}	Задачу *j*-й процессор принял	*pj_S_PT*
S_{SL}^{pj}	Выбран для обслуживания задачи *j*-й процессор	*en_sj*
S_{A}^{pj}	Выполняет обслуживание задачи *j*-й процессор	*pj_S_A*
S_{OF}^{t}	Удалить задачу из очереди	*t_S_OF*
S_{TO}^{pj}	Снять задачу с исполнения	*pj_S_TO*
S_{E}^{pj}	Выполнение задачи закончено	*pj_S_E*
S_{RT}^{t}	Выдача результата выполнения текущей задачи	*t_S_RT*
S_{O}^{pj}	*j*-й процессор занят	*not_enj*
S_{RE}^{pj}	Результат выполнения задачи выдан	*pj_S_RE*

На основании словесно представленного алгоритма управления процессами и введенных событий, реализуемых в этом алгоритме, система канонических уравнений, описывающих эти события, будет иметь следующий вид:

- для процесса «клиент», реализуемого диспетчером:

$$S_{Q}^{t}(t+1) = S_{I}^{t}\, S_{FQ}^{t} \vee S_{Q}^{t}\, \overline{S_{SL}^{pj}}; \qquad S_{ZPj}^{t}(t+1) = S_{Q}^{t}\, S_{SL}^{pj} \vee S_{ZPj}^{t}\, \overline{S_{PZ}^{pj}};$$

$$S_{GPj}^{t}(t+1) = S_{ZPj}^{t}\, S_{PZ}^{pj} \vee S_{GPj}^{t}\, \overline{S_{PT}^{pj}}. \qquad (1)$$

- для процесса «сервер», реализуемого процессором:

$$S_{S}^{p} = \overset{\vee}{_{j=1,N}}\, S_{S}^{pj}; \qquad S_{VP}^{pj}(t+1) = (S_{S}^{p}\, S_{ZP}^{t} \vee S_{O}^{pj})\, \overline{S_{SL}^{pj}}$$

$$S_{SL}^{pj}(t+1) = (\overset{\vee}{_{j=1,N}}\, S_{O}^{pj}\, S_{VZ}^{pj}\, S_{PR}^{pj}) \vee S_{SL}^{pj}\, \overline{S_{ZP}^{t}};$$

$$S_{PZ}^{pj}(t+1) = S_{SL}^{pj}\, S_{ZP}^{t} \vee S_{PZ}^{pj}\, \overline{S_{GPj}^{t}};$$

$$S_{PT}^{pj}(t+1) = S_{PZ}^{pj}\, S_{GPj}^{t}, \qquad (2)$$

где S_{VZ}^{pj} – комбинационное событие, обеспечивающее взаимоисключение при осуществлении процедуры выбора свободных процессоров для обслуживания очередной задачи на основе несовместимости события S_{SL}^{pj} с другими событиями из их общего числа, равного N, определяется выражением

$$S_{VZ}^{pj} = \overline{S_{SL}^{p_1}}\, \overline{S_{SL}^{p_2}} \ldots \overline{S_{SL}^{p_{j-1}}}\, \overline{S_{SL}^{p_{j+1}}} \ldots S_{SL}^{p_N}. \qquad (3)$$

S_{PR}^{pj} – событие, обеспечивающее приоритет выбора *j*-го процессора на обслуживание очередной задачи. Алгоритм вычисления приоритетного события для циклической дисциплины обслуживания дан в [7].

Система канонических уравнений, описывающая события после рандеву:

$$S_{A}^{pj}(t+1) = S_{PT}^{pj}\, S_{GPj}^{t} \vee S_{A}^{pj}\, \overline{S_{E}^{pj}}; \quad S_{TO}^{t}(t+1) = S_{A}^{pj}\, S_{E}^{pj} \vee S_{TO}^{t}\, \overline{S_{RT}^{pj}}$$

$$S_{RT}^{pj}(t+1) = S_{A}^{pj}\, S_{E}^{pj} \vee S_{RT}^{pj}\, \overline{S_{RE}^{pj}}; \; S_{OF}^{t}(t+1) = S_{TO}^{t}\, S_{RT}^{pj}; \qquad (4)$$

$$S_S^{pj}\,(t+1) = S_{RE}^{pj}\,S_{RT}^{pj} \vee S_S^{pj}\,\overline{S_{SL}^{pj}}.$$

Уравнениям соответствует граф НДА. На рис. 1 представлены графы клиентской и серверной частей алгоритма управления взаимодействующими процессами при диспетчеризации задач в многопроцессорной системе до момента «рандеву» (до оператора объединения J(&)) и часть графа после «рандеву» (после оператора объединения J(&)).

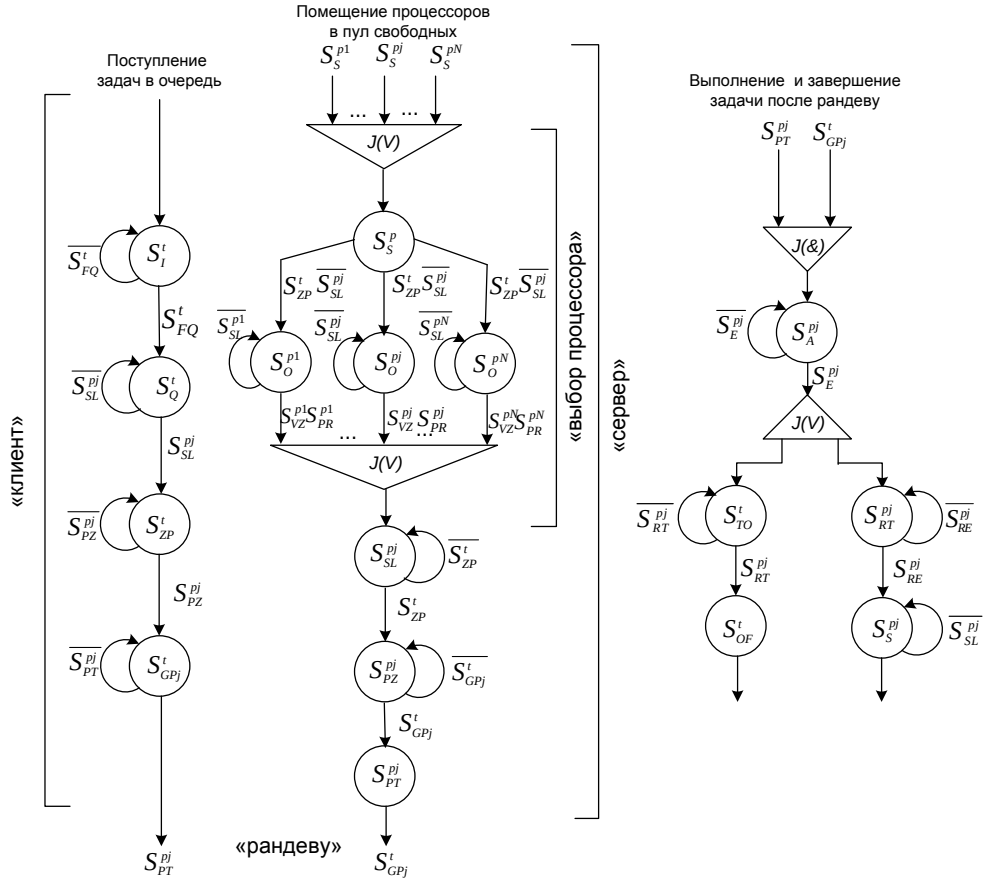

Рис. 1.

Граф НДА управления процессами при диспетчеризации задач

На основании модели НДА алгоритма управления диспетчеризацией задач строится модель цветной сети Петри. При построении используются следующие цвета: *BOOL*, *DATA*, *lDATA*, *PAK*. Для описания позиций и моделирующих сигналов (частные состояния цифрового автомата) использовался цвет *BOOL*. Фишки данного цвета могут принимать значения *true* и *false*. Как будет показано в дальнейшем, при использовании данного цвета описания функция передачи фишки в позицию сети Петри полностью аналогична уравнению перехода в новое состояние для цифрового автомата. С помощью цвета *DATA* моделируются задачи, поступающие в диспетчер. Данный цвет позволяет изменять назначение и количество полей, описывающих задачу [6]. Использование данного цвета позволяет при необходимости выполнять в процессе моделирования сбор статистики. Цвет *lDATA* используется при описании очереди задач в буфере типа *FIFO* (*first in first out*). Так как используемая сеть Петри не поддерживает ингибиторные дуги, мы используем цвет *PAK*. Позиции данного цвета могут содержать фишки двух цветов *DATA* или *AVAIL*. Если в позиции находится фишка цвета *AVAIL*, то это означает, что данная позиция свободна.

Модель устройства диспетчеризации задач, включающая операционную часть и управляющий автомат, представлена на рис. 2.

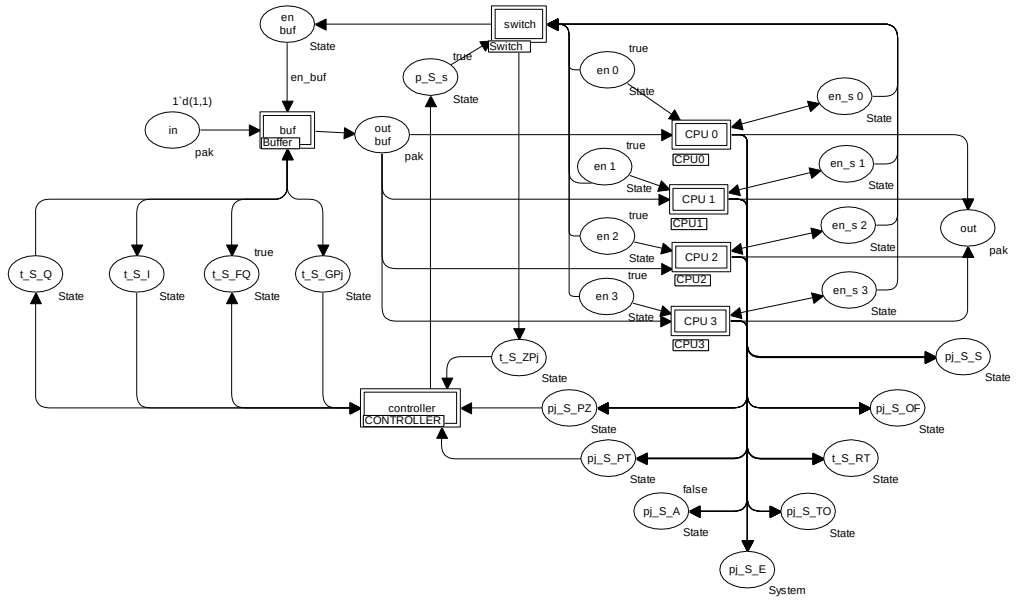

Рис. 2. Модель устройства диспетчеризации

Она содержит: входной порт (*in*), через который поступают задачи; выходной порт (*out*), через который процессоры выдают результаты выполнения задачи; очередь задач (*buf*), которая строится по принципу буфера *FIFO*; узел выбора (*switch*), определяющий процессор для обработки очередной задачи в соответствии с приоритетной дисциплиной обслуживания и циклическим сдвигом приоритетов; четыре процессора (*CPU0, CPU1, CPU2, CPU3*), которые моделируют алгоритм выполнения задачи; сигнальные позиции, через которые узлы обмениваются информацией; узел управления (*controller*), представляющий собой управляющий автомат.

На рис. 3 представлена модель буфера задач. При поступлении новой задачи переходом *in* формируется сигнал, по которому при наличии свободного места задача помещается в буфер. Если в очереди отсутствуют свободные места, то задача помещается в позицию *Garbage collector* и далее не обслуживается. Данная позиция необходима для сбора статистики о необслуженных задачах. При постановке задачи в очередь проверяется, остаются ли при этом свободные места. Если свободных мест нет, то формируется сигнал о заполнении очереди. Если при заполненной очереди была выполнена операция изъятия задачи, то формируется сигнал о наличии свободного места. При изъятии задачи из очереди формируется сигнал о готовности её к исполнению.

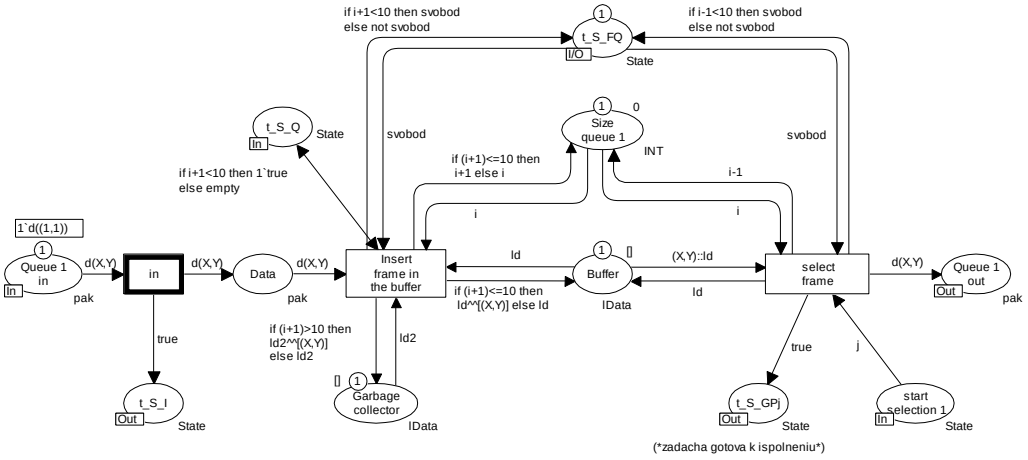

Рис. 3. Модель буфера задач

В представленной работе разработаны также модели узла выбора процессора, процессорных узлов и узла управления *controller* (на рисунках не представлены).

Поскольку подсети, моделирующие работу процессорных узлов, идентичны, поэтому представлена модель только одного процессора. Модели других процессорных узлов включаются аналогично. Разработка модели велась по следующему алгоритму. При получении сигнала запроса процессор формирует сигнал подтверждения запроса на выполнение задачи. Изъятая из буфера задача поступает в процессор, где формируется сигнал о её принятии на выполнение. Одновременно с этим процессор изменяет флаг своего состояния, который указывает на занятость процессора, и начинает выполнять задачу. По окончании обслуживания процессор формирует сигнал завершения, после чего он последовательно формирует сигналы снятия задачи с выполнения, выдачи результата, включения свободного процессора в пул, и удаления задачи, после чего флаг состояния процессора изменяется, указывая на освобождение процессора.

Узел управления *controller* описывает последовательность формирования сигналов конечным автоматом, используя метод синхронизации «рандеву» для параллельных процессов, реализуемый клиентом и сервером.

Заключение

В данной работе предложен метод сетей Петри с использованием программного пакета CPN TOOLS для визуализированной отладки алгоритма, реализуемого в устройстве аппаратной поддержки диспетчеризации задач в составе четырехпроцессорной системы, без построения самой аппаратуры. Кроме того, предложенный метод может служить основой для оценки показателей производительности, например времени ответа устройства.

СПИСОК ЛИТЕРАТУРЫ

1. Таненбаум Э. Современные операционные системы. – 3-е изд. – СПб. : Питер, 2010. – 1120 с.

2. Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования : Пер. с англ. – М. : Вильямс, 2003. – 512 с.

3. Вашкевич Н.П. Аппаратная поддержка планировщика задач с глобальной очередью в многопроцессорных системах / Н.П. Вашкевич, Р.А. Бикташев, А.И. Меркурьев // Известия вузов. Поволжский регион. – 2011. – № 1. – (Технические науки).

4. Jensen K., Kristensen L. Coloured Petri Nets: modelling and validation of concurrent systems. Springer-Verlag, 2009.

5. Вашкевич Н.П. Недетерминированные автоматы в проектировании систем параллельной обработки : Учеб. пособие / Н.П. Вашкевич. – Пенза : Изд-во Пенз. гос. ун-та, 2004. – 280 с.

6. Механов, Коннов Н.Н., Кизилов Е.А. Построение сети Петри, моделирующей поведение недетерминированного конечного автомата // Труды XIX Всероссийской научно-методической конференции «Телематика'2012». – СПб. : ИТМО, 2012.

7. Вашкевич Н.П. Аппаратная реализация функций синхронизации параллельных процессов при обращении к разделяемому ресурсу на основе ПЛИС / Н.П. Вашкевич, Р.А. Бикташев, Е.И. Гурин // Известия вузов. Поволжский регион. – 2007. – № 2. – С. 3–12. – (Технические науки).