УДК 681.3

Д.Б. Борзов, д-р техн. наук, профессор, ФГБОУ ВО «Юго-Западный государственный университет» (Россия, 305040, Курск, ул. 50 лет Октября, 94) (e-mail: borzvdb@kursknet.ru)

И.И. Масюков, аспирант, ФГБОУ ВО «Юго-Западный государственный университет» (Россия, 305040, Курск, ул. 50 лет Октября, 94) (e-mail: ilmas46ru@gmail.com)

ПЛАНИРОВАНИЕ ЗАГРУЗКИ ПРОЦЕССОРОВ В МУЛЬТИПРОЦЕССОРНЫХ СИСТЕМАХ КРИТИЧЕСКОГО НАЗНАЧЕНИЯ

В настоящее время широко используются мультипроцессорные системы критического характера. Такие системы применяются при слежении, прицеливании, наблюдении и т.п. Подобные задачи, как правило, требуют максимального увеличения производительности и уменьшения времени решения задачи. Для этих целей используется первоначальное выделение независимых линейных, условных и циклических участков последовательных программ [1]. Это выполняется для высвобождения фрагментов программ, которые возможно назначать для исполнения на процессоры таким образом, чтобы при исполнении они как можно меньше обменивались данными с соседними процессорами [2]. За счет этого возможно частичное повышение производительности мультипроцессорной вычислительной системы вместе с уменьшением общего времени выполнения всей задачи в целом.

Для систем рассматриваемого характера процессора всей системы желательно назначать фрагментами программ так, чтобы они были постоянно загружены на протяжении решения всей задачи. Это является другим способом повышения производительности мультипроцессорной системы. Очевидно, что использования для этих целей программных средств нереально из-за критичности временного параметра. Следовательно, актуальным является использование методов и соответствующих аппаратно-ориентированных алгоритмов планирования загрузки процессоров, что является предметом исследований в данной статье.

В статье показана актуальность постоянной загрузки процессоров мультипроцессорных систем с высоким коэффициентом готовности. Обоснована необходимость составления плана загрузки процессоров для поддержания этого коэффициента. Предложен соответствующий метод и алгоритм для мультипроцессорных систем критического назначения (системы слежения, наблюдения, прицеливания и т.п.).

Ключевые слова: *теория расписаний хост-процессор; мультипроцессорные системы; параллельная обработка; системы высокой готовности; критические системы.*

DOI: 10.21869/2223-1560-2018-22-6-168-174

Ссылка для цитирования: Борзов Д.Б., Масюков И.И. Планирование загрузки процессоров в мультипроцессорных системах критического назначения // Известия Юго-Западного государственного университета. 2018. Т. 22, № 6(81). С. 168-174.

***

Введение

В критических мультипроцессорных системах (наблюдение, слежение, прицеливания, управляющие системы и т.п.) часто возникает необходимость моделирования решаемых задач. Применение для этих целей программных средств неприемлемо из-за временных ограничений, накладываемых мультипроцессорной системой и прогнозируемого роста аппаратной сложности мультипроцессорной системы при увеличении времени выполняемых задач [1-3].

Наибольшее распространение получило несколько алгоритмов планирования, к которым относятся алгоритмы «первым пришел, первым обслужен» и его модификация, в которой все процессы построены по циклической схеме, алгоритм гарантированного планирования и приоритетного планирования.

Описанные подходы являются узкоспециализированными и не подходят для применения в мультипроцессорных системах критического назначения. Вследствие этого, актуальным является поиск

соответствующего метода и алгоритма планирования загрузки.

Алгоритмы, соответствующие методы и устройства распараллеливания последовательных программ [4-6], как правило, не адаптированы для мультипроцессорных систем критического характера в связи с отсутствием учета числа процессоров мультипроцессорной системы, игнорированием приоритета команд и т.п. Более того, это приводит к увеличению времени выполнения всей задачи в целом и уменьшению производительности системы.

Таким образом, можно говорить об актуальности разработки метода, алгоритмов, ориентированных на их аппаратную реализацию, и соответствующих специализированных устройств планирования загрузки процессоров.

На основе вышесказанного можно сделать вывод об актуальности разработки метода и алгоритмов планирования загрузки процессоров. Работа основана на исследованиях [4-7].

Постановка задачи

Введем ряд обозначений и определений.

Проектом будем называть загрузку процессоров, ограниченных связностью данных работ. В данном случае задачей является построение расписания выполнения работ проекта с учетов возможности предшествования и ограничений на ресурсы (Resource-Constrained Project Scheduling Problem, RCPSP).

Алгоритм распределения построен на основе идей метода RCPSP теории расписаний (рис. 1) [8-10].

Алгоритм, представленный на рисунке 1а, состоит из трех операторов: 1, 2 и 3. Хост–компьютером предусмотрен оператор 4, который должен быть выполнен вне зависимости от каких-либо условий.

Трудоемкость вычислений сопоставляется с индивидуальным оператором. Это значение проставлено цифрой сверху над кружком с номером оператора.

В суммарной загрузке процессора четвертый оператор вычисляется индивидуально. Два варианта расписаний процессоров изображены на рисунках 1б и 1в. Расписание, показанное на рисунке 1б, равно одиннадцати и является неоптимальным. Данный вариант улучшается до девяти, как показано на рисунке 1в. Это возможно выполнить с помощью предложенного в данной статье метода плана загрузки процессоров.

Пусть даны задачи $\{G_{nm}\} = \{1,\ldots,\mathrm{n}\}$ и P обновляемых ресурсов (процессоров) где $k = 1,\ldots,\mathrm{P}$. Считается, что в каждый момент времени t предоставляется P_k единиц ресурса k.

Пусть первоначально задана продолжительность обслуживания $d_i \geq 0$ для любой задачи $i = 1,\ldots,n$. В течение обработки задачи i необходимо $q_{ik} \leq P_k$ единиц ресурса $k = 1,\ldots,\mathrm{P}$. После окончания решения всех задач, свободные в данный момент процессоры могут быть назначены на выполнение других.

Считаем, что пары задач связаны между собой соотношением предшествования: i → j означает, что обработка задачи j начинается после окончания обработки задачи i на ветви Br.

Считаем, что задачи обрабатываются с момента времени t = 0 и при этом прерывание на обработку других задач запрещена.

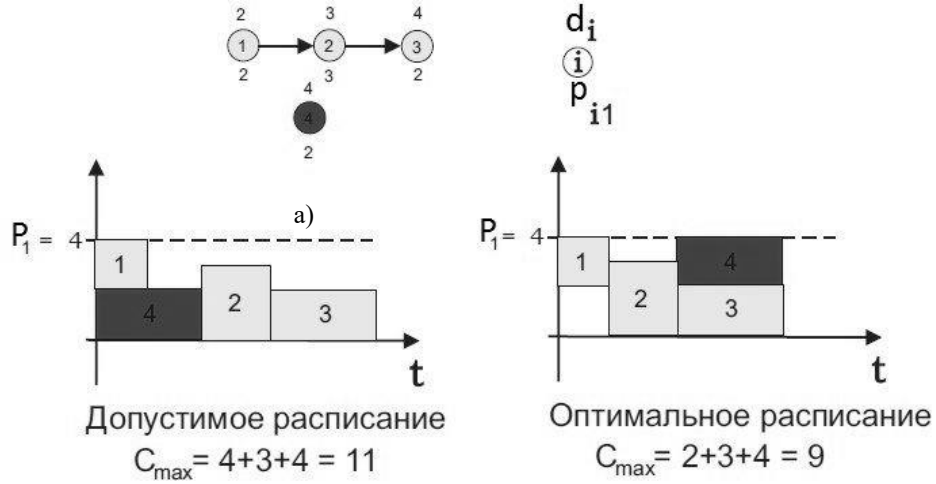

Рис. 1. Пример задачи составления оптимально расписания

Пусть необходимо определить время начала обработки задач S_i $(i=1,...,n)$ таким образом, чтобы $C_{\max}=\max\limits_{i=1,...,n}\{C_i\}$ $\left(C_i=S_i+d_i\right)$ с учетом выполнения ограничений:

1) необходимо выполнение в произвольное время $t\in[0,\mathrm{C}_{\max})$ условия $\sum\limits_{i:=1}^{n} g_{ik}\varphi_i(\mathrm{t})\le P_k,\mathrm{k}=1,...,P$, где $\varphi_i(\mathrm{t})=1$, если i-я задача обрабатывается в t-е время и $\varphi_i(\mathrm{t})=0$ – в альтернативном случае. Таким образом, задачи в процессе вычисления необходимы быть обеспечены процессором P;

2) между задачами не должно быть нарушено отношение предшествования, т.е.

$S_i+d_i\le S_j$, если i $\to$ j для i, $j\in G$.

Происходит распараллеливание для любого яруса Yar, из-за чего зависимость от Br является минимальной.

Математическая модель и алгоритм

Представим множество $\{G_{nm}\}$ как входную матрицу команд, состоящую из Yar и Br. $\{S_{so}\}$ состоит из $\mathrm{Yar_i}$, t_{ij} – плановый срок (момент времени, к которому i-я задача должна быть выполнена), $\{S^k{}_{so}\}$ - множество $\{S_{so}\}$, ограниченное количеством процессоров $\{P\}$. Подмножество $\{S^k{}_{p_i}\}$ устанавливает расписание команд для определенного P, $\{G'_{nm}\}$ – содержит план загрузки процессоров в мультипроцессорных критических системах.

Таким образом, алгоритм может быть описан следующей последовательностью шагов:

1. Получить граф YB.

2. Преобразовать графYB в множество $\{G_{nm}\}$.

3. Разбить $\{G_{nm}\}$ на подмножества $\{S_{so}\}$.

4. Отсортировать операции в подмножестве $\{S_{so}\}$ по уменьшению t_{ij}.

5. Разбить подмножества $\{S_{so}\}$ на $\{S^k{}_{so}\}$, где $k=1...m$, m ограничение на количество процессоров.

6. Установить соответствие операций определенному процессору $\{S^k{}_{p_i}\}$, $k=1...m$.

7. На каждом временном отрезке множество $\{S_{so}\}$ отсортировать по возрастанию t_{ij}.

8. При $t = 0$ для процессора m загрузить операцию подмножества $\{S^{k}{}_{p_i}\}$.

9. Записать в $\{G'_{nm}\}$ новый результат.

10. Повторять п. 5-9 пока $\{S_{so}\} \neq \varnothing$.

Конец.

Моделирование представленного метода и алгоритма

Проведено моделирование разработанного метода и алгоритма с целью анализа скорости обработки задач на процессорах мультипроцессорной системы (рис. 2) и увеличения количества процессоров при определенном количестве информации для одной последовательной ветви алгоритма задачи (рис. 3). Исследования проходили при исходных данных Yar = {1…20} и Br = {1…20}. Выбраны длительности операций для моделирования t={1…16}, которые присваиваются произвольно.

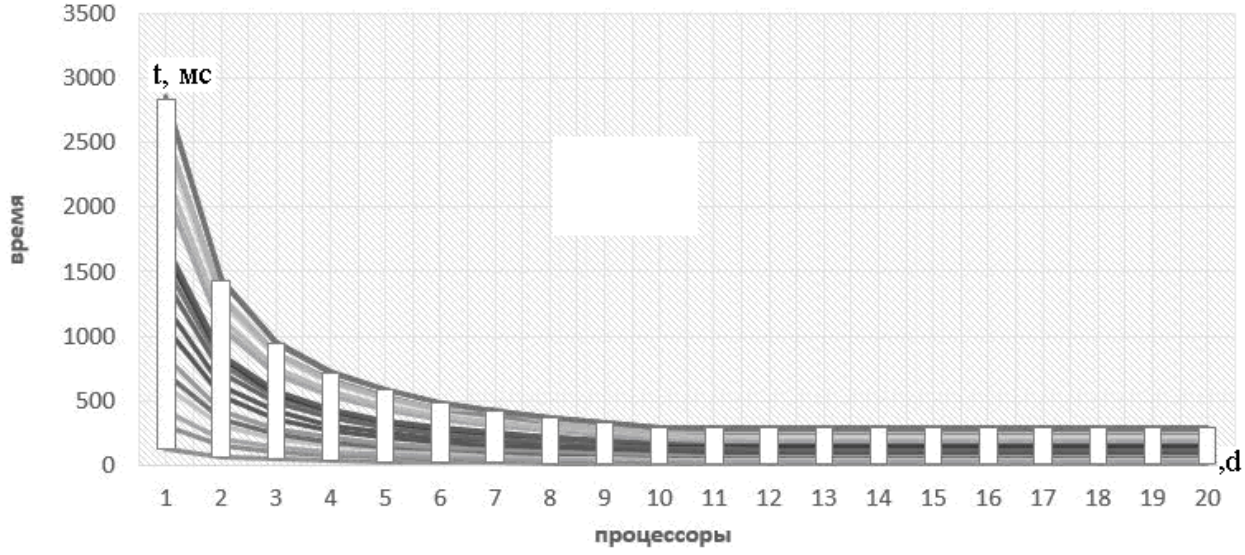

Рис. 2. График загрузки процессоров

График (см. рис. 2) показывает изменение времени $d = \sum_{i=1}^{n} t_i$ в зависимости от количества процессоров. Анализ зависимости показывает, что изменение показателя $d = [2;4]$ уменьшает время выполнения в 5 раз, по сравнению с $d = [1;2]$ при последовательной обработке данных. Активность $d = [10;20]$ показывает наступление равноденствия во времени, причем дальнейшее увеличение процессоров не валяет на производительность мультипроцессорной системы. Это доказывает справедливость закона Амдала [7], который гласит, что решаемая задача с адаптированным для нее числом процессоров остается неизменной.

Программный код решаемой задачи состоит из двух частей: последовательной и распараллеливаемой. Доля операций, которые должны выполнятся последовательно, обозначены f, где $0 \leq f \leq 1$. Доля, приходящаяся для распараллеливания, обозначатся $1 - f$ (f=0, если задачи распараллелены, f=1, если данные последовательны). Распараллеливаемая часть программы равномерно распределяется по всем процессорам.

С учетом этого имеем:

$$I_p = f \times I_s + \frac{(1-f) \times I_s}{n}. \qquad (1)$$

В результате (1) получается формулировка Амдала, выражающая ускорение, которое может быть достигнуто для n процессоров:

$$S = \frac{T_s}{T_p} = \frac{n}{1+(n-1)\times f}. \quad (2)$$

Если в (2) принять $n \to \infty$, получим:

$$\lim_{n\to\infty} s = \frac{1}{f}. \quad (3)$$

Из (3) следует, что если в программе 10% последовательных операций (f=0,1), то при любом числе процессоров, ускорение в 10 раз невозможно при пределе теоретической оценки, равной 10.

Распараллеливание ведет к издержкам, связанным с распределением задач по процессорам, и составление такого плана загрузки может загрузить систему так, чтобы освободить ресурсы для выполнения других задач.

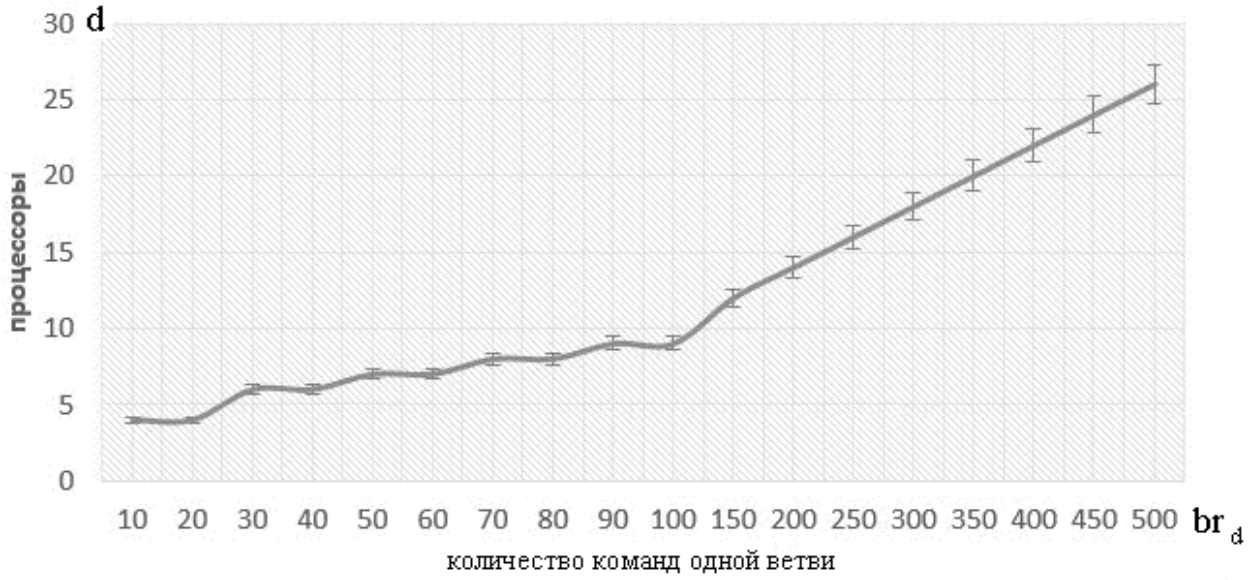

Рис. 3. График влияния команд одной ветви на количество процессоров

График (см. рис. 3) показывает необходимое и достаточное число процессоров для обработки определенного количества информации. Из анализа рисунка 4 можно заключить, что при изменении количества команд с 10 до 500, увеличение количества процессоров возрастает примерно в 6 раз, что является необходимым фактором для оптимальной загрузки.

Выводы

Задача поиска необходимого и достаточного количества процессоров для обработки информации стоит в оптимизации загрузки процессоров на 70-80%. Данное исследование и предложенный метод могут быть использованы для процессоров критических мультипроцессорных систем высокой готовности. Ресурсы хост–процессора могут быть использованы для поиска оптимально количества процессоров, необходимого для обработки информации, и при построении расписания исходя из ограниченности мультипроцессорных систем.

Основываясь на анализе проблемной области [7-10], можно сделать вывод, что задача оптимизации и ускорения работы процессоров в критических системах актуальна. Для решения задачи планирования загрузки процессоров в мультипроцессорной системе, в дальнейшем разра-

ботанные алгоритмы могут быть взяты за основу при построении специализированного вычислительного устройства [12], позволяющего оптимизировать как общее время исполнения, так и подобрать необходимое количество процессоров для максимальной нагрузки.

Список литературы

1. Воеводин В.В. Параллельные вычисления. СПб.: БХВ – Петербург, 2002. 608 с.

2. Трахтенгерц Э.А. Введение в теорию анализа и распараллеливания программ ЭВМ в процессе трансляции. М.: Наука,1981.254 с.

3. Корнеев В.В. Параллельные вычислительные системы. М.: Нолидж, 1999. 340 с.

4. Дюбрюкс С.А., Борзов Д.Б. Метод выявления параллелизма внутри линейных участков последовательных программ и его аппаратная реализация // Известия вузов. Приборостроение. 2008. Т.2. С. 34–38.

5. Ткачев П.Ю., Борзов Д.Б. Метод и алгоритм поиска линейных участков внутри циклов с последующем распараллеливанием // Известия Юго-Западного государственного университета. 2015. №5(62). С. 16-21.

6. Борзов Д.Б., Дюбрюкс С.А., Титов В.С. Выявление параллелизма внутри линейных участков последовательных программ со связями по управлению // Машиностроение и техносфера XXI века: сборник трудов XIV Международной научно–технической конференции. Т2. Донецк, 2007. С. 26–30.

7. Бабаян Б.А., Бочаров А.В. Многопроцессорные ЭВМ и методы их проектирования / под ред. Ю.М. Смирнова. М.: Высшая школа, 1990. 142 с.

8. Танаев В.С., Ковалев М.Я. Теория расписаний. Групповые технологии. Минск: Институт технической кибернетики НАН Беларуси, 1998. 290 с.

9. Лазарев А.А., Гафаров Е.Р. Теория расписаний: задачи и алгоритмы. М., 2011.С.31-32.

10. Конвей Р.В., Максвелл В.Л., Миллер Л.В. Теория расписаний. М.: Наука, 1975. 365 с.

11. Антонов А. Под законом Амдала (рус.) // Компьютерра. 11.02.2002. № 430.

12. Амамия М., Танака Ю. Архитектура ЭВМ и искусственный интеллект. М.: Мир, 1993. 400 с.

Поступила в редакцию 26.11.18

UDC 681.3

D.B. Borzov, Doctor of Engineering Sciences, Professor, Southwest State University
(Russia, 305040, Kursk, 50 Let Oktyabrya str., 94) (e-mail: borzovdb@kursknet.ru)

I.I. Masyukov, Post-Graduate Student, Southwest State University
(Russia, 305040, Kursk, 50 Let Oktyabrya str., 94) (e-mail: ilmas46ru@gmail.com)

PLANNING OF DOWNLOAD PROCESSORS IN MULTIPROCESSOR SYSTEMS OF CRITICAL PURPOSE

At present, multiprocessor systems of a critical nature are widely used. Such systems are used for tracking, aiming, observing, etc. Such tasks, as a rule, require maximizing productivity and reducing the time to solve a problem. For these purposes, the initial selection of non-dependent linear, conditional and cyclic sections of sequential programs is used [1]. This is done to release fragments of programs that can be assigned to execution on processors in such a way that during execution they exchange data with neighboring processors as little as possible

[2]. Due to this, it is possible to partially improve the performance of a multiprocessor computing system, together with a decrease in the overall execution time of the entire task as a whole.

For systems of the considered nature of the processor of the entire system, it is desirable to assign program fragments so that they are constantly loaded throughout the solution of the entire problem. This is another way to improve the performance of a multiprocessor system. It is obvious that the use of software for this purpose is not real due to the criticality of the time parameter. Therefore, it is relevant to use methods and corresponding hardware-oriented algorithms for scheduling processor loads, which is the subject of research in this article.

The article shows the relevance of the constant loading of processors of multiprocessor systems with a high availability factor. The necessity of drawing up a plan for loading processors to support this coefficient is substantiated. An appropriate method and algorithm for multiprocessor systems for critical purposes (tracking systems, surveillance, aiming, etc.) are proposed.

Key words: *scheduling theory; host processor; multiprocessor systems; parallel processing; high availability systems; critical systems.*

DOI: 10.21869/2223-1560-2018-22-6-168-174

For citation: Borzov D.B., Masyukov I.I. Planning of Download Processors in Multiprocessor Systems of Critical Purpose. Proceedings of the Southwest State University, 2018, vol. 22, no. 6(81), pp. 168-174 (in Russ.).

***

Reference

1. Voevodin V.V. Parallel'nye vychislenija. Saint-Petersburg, BHV – Peterburg Publ., 2002, 608 p.

2. Trahtengerc Je.A. Vvedenie v teoriju analiza i rasparallelivanija programm JeVM v processe transljacii. Moscow, Nauka Publ., 1981, 254 p.

3. Korneev V.V. Parallel'nye vychislitel'nye sistemy. Moscow, Nolidzh Publ., 1999, 340 p.

4. Djubrjuks S.A., Borzov D.B. Metod vyjavlenija parallelizma vnutri linejnyh uchastkov posledovatel'nyh programm i ego apparatnaja realizacija. *Izvestija vuzov. Priborostroenie*, 2008, vol. 2, pp. 34–38.

5. Tkachev P.Ju., Borzov D.B. Metod i algoritm poiska linejnyh uchastkov vnutri ciklov s posledujushhem rasparallelivaniem. *Izvestija Jugo-Zapadnogo gosudarstvennogo universiteta*, 2015, no.5(62), pp. 16-21.

6. Borzov D.B., Djubrjuks S.A., Titov V.S. Vyjavlenie parallelizma vnutri linejnyh uchastkov posledovatel'nyh programm so svjazjami po upravleniju. Mashinostroenie i tehnosfera XXI veka. Sbornik trudov XIV Mezhdunarodnoj nauchno–tehnicheskoj konferencii. Doneck, 2007, vol. 2. pp. 26–30.

7. Babajan B.A., Bocharov A.V. Mnogoprocessornye JeVM i metody ih proektirovanija; ed. by Smirnov Ju.M. Moscow, Vysshaja shkola Publ., 1990, 142 p.

8. Tanaev V.S., Kovalev M.Ja. Teorija raspisanij. Gruppovye tehnologii. Minsk, 1998, 290 p.

9. Lazarev A.A., Gafarov E.R. Teorija raspisanij: zadachi i algoritmy. Moscow, 2011, pp.31-32.

10. Konvej R.V., Maksvell V.L., Miller L.V. Teorija raspisanij. Moscow, Nauka Publ., 1975, 365 p.

11. Antonov A. Pod zakonom Amdala (rus.). *Komp'juterra*. 11.02.2002, no. 430.

12. Amamija M., Tanaka Ju. Arhitektura JeVM i iskusstvennyj intellekt. Moscow, Mir Publ., 1993, 400 p.