ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № ?, 2014, с. ??–??

УДК 519.687.1

Р. А. Ахметшин, А. В. Юлдашев

Распределение многопоточных программ на многопроцессорных системах с неоднородным доступом к памяти

Аннотация. В работе построена модель подсистемы памяти для оценки временных задержек при выполнении многопоточных программ на современных многопроцессорных системах с неоднородным доступом к памяти. Предложен алгоритм распределения задач на кластерной системе, учитывающий конкуренцию за доступ к подсистеме памяти современных вычислительных узлов. Данный алгоритм реализован на основе построенной модели доступа к памяти. Экспериментально показана возможность снижения времени выполнения программ на кластерной системе при использовании предложенного алгоритма распределения задач по узлам кластерной системы.

Ключевые слова и фразы: многопроцессорные системы, кластерные системы, высокопроизводительные вычисления, системы с неоднородным доступом к памяти.

Введение

При выполнении многопоточных программ на современных многопроцессорных системах возникает конкуренция между потоками за общие ресурсы вычислительной системы. Например, при выполнении приложений, интенсивно использующих оперативную память, возникает конкуренция за каналы доступа к памяти, которая может существенно снизить производительность вычислений. Для поддержания высокой производительности вычислений необходимо учитывать нагрузку на оперативную память системы. Также необходимо учитывать то, что доступ к памяти на современных многопроцессорных системах неоднороден (Non-Uniform Memory Access, NUMA), то есть время доступа к разным областям данных, адресуемых в многопоточной программе, может различаться.

В данной работе предложен алгоритм распределения задач (многопоточных программ) по узлам кластерной системы с учетом особенности организации памяти в современных многопроцессорных вычислительных узлах.

Для реализации алгоритма, приведенного в работе, построена модель доступа к подсистеме памяти в двухпроцессорных NUMA-системах. Модель позволяет оценить замедление времени выполнения в конкурентном режиме многопоточной программы, интенсивно использующей оперативную память.

На примере программного комплекса численного моделирования нефтегазовых месторождений экспериментально показано, что использование предложенного подхода к распределению задач на многопроцессорных системах позволяет снизить среднее время выполнения разнородного по характеристикам набора многопоточных программ.

1. Модель подсистемы памяти двухпроцессорной NUMA-системы

В качестве моделируемого объекта рассмотрим двухпроцессорную систему на базе процессоров Intel Xeon E5-2670. Каждый из двух процессоров имеет контроллер памяти (КП), обозначим их КП1 и КП2, через которые осуществляется доступ к двум областям данных, образующих единое адресное пространство вычислительной системы. Процессоры соединенны между собой специальной шиной QPI (Quick Path Interconnect), пропускная способность которой ниже чем у КП. Такое разделение памяти и обеспечивает эффект неоднородного доступа.

Можно выделить следующие источники замедление выполнения многопоточных приложений на современных NUMA-системах, возникающих при работе с различными уровнями иерархии памяти: конкуренция между потоками одной или нескольких программ за кэш-память последнего уровня, которая является общей для ядер одного процессора; конкуренция, возникающая на КП; конкуренция при использовании шины QPI; задержка, возникающая при обращении к удаленной области памяти. Исследования, проведенные в работе [1], показали, что основной вклад в замедление выполнения приложений в конкурентном режиме (70–85 %) оказывает конкуренция, возникающая при использовании КП и шины QPI. В следствии выше сказанного, в нашей модели доступа к подсистеме памяти NUMA-

систем учитываются временные задержки, возникающие только на этих устройствах (КП1, КП2 и QPI).

Построим модель подсистемы памяти NUMA-систем с использованием теории массового обслуживания [2]. Для анализа временных задержек представим нашу модель в виде сети систем массового обслуживания (СМО). Схема сети представлена рисунке 1.

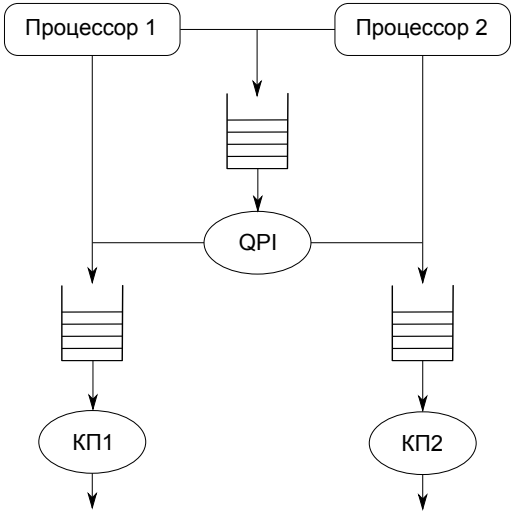

Рис. 1. Схема сети СМО

Функционирование каждого устройства (КП1, КП2 и QPI) опишем СМО с одним обслуживающим прибором, пуассоновским потоком входящих заявок и показательным законом распределения времени обслуживания (M/M/1).

Обозначим через $\lambda_{\text{КП1}}$, $\lambda_{\text{КП2}}$, λ_{QPI} интенсивности поступления заявок в устройства КП1, КП2 и QPI соответственно, а через $\mu_{\text{КП1}}$, $\mu_{\text{КП2}}$, μ_{QPI} интенсивности обслуживания заявок на тех же приборах. Причем, $\mu_{\text{КП1}} = \mu_{\text{КП2}} = \mu$. Запишем интенсивность входящего потока для каждого устройства:

$$\lambda_i = \frac{N_i}{T}, \qquad i \in \{\text{КП1}, \text{КП2}, QPI\}, \tag{1}$$

где N_i — количество заявок, обслуженное i-тым устройством,

T — время работы многопоточной программы.

В свою очередь время работы приложения представим в следующем виде:

$$T = t + n_{\text{КП1}} t_{\text{КП1}} + n_{\text{КП2}} t_{\text{КП2}} + n_{QPI} t_{QPI}, \tag{2}$$

где t — время работы программы за исключением времени, необходимого для доступа к оперативной памяти,
$n_{\text{КП1}}$, $n_{\text{КП2}}$, n_{QPI} — количество заявок, поступивших от одного потока в СМО КП1, КП2 и QPI соответственно,
$t_{\text{КП1}}$, $t_{\text{КП2}}$, t_{QPI} — среднее время нахождения заявки в СМО КП1, КП2 и QPI соответственно.

Для систем M/M/1 при условии стационарности [2]:

$$\lambda_i < \mu_i, \qquad i \in \{\text{КП1}, \text{КП2}, QPI\}, \tag{3}$$

среднее время пребывания заявки в системе запишется следующим образом [2]:

$$t_i = \frac{1}{\mu_i - \lambda_i}, \qquad i \in \{\text{КП1}, \text{КП2}, QPI\}. \tag{4}$$

Если мы знаем доли заявок, поступивших на каждое устройство: $K_{\text{КП1}}$, $K_{\text{КП2}}$, K_{QPI}, от общего числа обращений к оперативной памяти N, которая генерирует многопоточное приложение, а также информацию о том, как соотносится общее число обращений N и характер обращений к подсистеме памяти отдельного потока, то, используя уравнения (1), (2) и (4), можно составить следующую систему уравнений:

$$\lambda_i = \frac{K_i}{\beta + \frac{k_{\text{КП1}}}{\mu - \lambda_{\text{КП1}}} + \frac{k_{\text{КП2}}}{\mu - \lambda_{\text{КП2}}} + \frac{k_{QPI}}{\mu_{QPI} - \lambda_{QPI}}}, i \in \{\text{КП1}, \text{КП2}, QPI\}, \tag{5}$$

где $\beta = \frac{t}{N}$ — величина, характеризующая интенсивность обращений к оперативной памяти многопоточной программы,
$k_i = \frac{n_i}{N}$.

Таким образом, зная характеристики программы: $k_{\text{КП1}}$, $k_{\text{КП2}}$, k_{QPI} и β, можно решить систему (5), тем самым, найти интенсивность потока заявок на каждом устройстве. При этом, решение системы (5) должно удовлетворять условию (3). Используя полученное

решение, можно найти величину задержек, возникающих на каждом устройстве, подставив найденное решение в уравнения (4).

2. Распределение потоков по ядрам

В современных операционных системах существует возможность привязки исполнения потоков к конкретным ядрам (процессорам) вычислительной системы. Это позволяет исключить миграцию потоков между вычислительными ядрами (процессорами), а также облегчить оптимальное распределение потоков с точки зрения доступа к подсистеме памяти, что играет важную роль в случае NUMA-систем.

Можно выделить два противоположных режима распределения потоков по ядрам вычислительного узла:

(1) потоки располагаются как можно ближе в соответствии с топологией вычислительного узла (compact);
(2) потоки располагаются как можно дальше в соответствии с топологией вычислительного узла (scatter).

Компилятор Intel обеспечивает возможность использования приведенных режимов распределения потоков для OpenMP программ путем объявления соответствующих переменных окружения [**3**].

Чтобы выбрать наилучший режим распределения потоков, при котором время выполнения программы будет наименьшим, можно воспользоваться предложенной моделью подсистемы памяти. Однако необходимо знать характеристики работы приложения с подсистемой памяти, общий подход к оценке которых изложен далее.

3. Априорная оценка характеристик поступающих задач

Характеристики работы приложения с подсистемой памяти, как правило, до запуска задачи неизвестны. В то же время можно измерить эти характеристики во время расчета задач средствами системных профилировщиков и с учетом того, что на кластерных системах часто запускается множество схожих задач, использовать данную информацию в последующем для априорной оценки характеристик вновь поступающих задач на основе набранной статистики.

Процедура априорной оценки характеристики соответствующих многопоточных программ, проводится нами в два этапа:

(1) объединение выполненных ранее задач в группы (кластеризация) и расчет усредненных характеристик по группам;

(2) выбор наиболее подходящей группы для новой задачи с неизвестной характеристикой (классификация).

На первом этапе априорной оценки проводится объединение схожих между собой задач в группы на основе алгоритма иерархической кластеризации [4]. Проведенные эксперименты показали, что в нашем случае наиболее качественная кластеризация достигается при использовании методов полной и средней связи, в которых расстояние между кластерами определяется как максимальное и среднее между элементами двух кластеров соответственно.

На втором этапе решается задача поиска наиболее близкого кластера для новой задачи с неизвестным значением интенсивности обращений к оперативной памяти. Расстояние до кластера рассчитывается как среднее расстояние до всех элементов кластера. В результате новая задача относится к наиболее близкому кластеру, из сформированных ранее, и ей ставится в соответствие среднее значение характеристики работы с подсистемой памяти в выбранном кластере.

4. Алгоритм распределения задач по узлам кластерной системы с учетом конкуренции за доступ к оперативной памяти вычислительных узлов

Для реализации алгоритма распределения задач по узлам кластера, учитывающего конкуренцию на узлах кластерной системы за доступ к оперативной памяти, можно использовать описанную ранее модель подсистемы памяти. А именно, для вновь прибывшей задачи выбирается тот узел, на котором замедление работы приложения, рассчитываемое с использованием модели, будет минимально.

При поступлении новой задачи необходимо выбрать определенный узел. Пусть нам известны характеристики задач, запущенных на каждом узле, и характеристики задачи, которую необходимо запустить. Номер узла n будет определяться решением следующей задачи минимизации:

$$n = \arg\min_i \frac{T_i^{cont}}{T^{clean}}, \qquad i = \overline{1, N}, \tag{6}$$

где T_i^{cont} — время выполнения новой задачи на i-том узле с учетом конкуренции,
T^{clean} — время выполнения новой задачи на свободном узле,
N — число узлов кластера.

5. Апробация предложенного алгоритма

Тестирование предложенного алгоритма проводилось на примере распределения задач (многопоточных программ) гидродинамического моделирования нефтегазовых месторождений с различными входными данными на вычислительном кластере, состоящем из двухпроцессорных узлов на базе Intel Xeon E5-2670. Тестовый набор содержит восемь задач, различающихся по времени выполнения и интенсивности работы с оперативной памятью. Все задачи запускались на восьми ядрах каждая в режиме распределения потоков по ядрам scatter. Таким образом, в конкурентном режиме на одном узле одновременно могли считаться по две задачи.

Сравнивалось три варианта запуска тестового набора:

(1) сначала задачи запускались по одной на узле (этот вариант необходим для сравнения с двумя другими вариантами);
(2) следующий вариант, описывает наихудший способ запуска, где задачи наиболее интенсивно использующие оперативную память размещались на одних и тех же узлах;
(3) и, наконец, вариант запуска тестового набора задач, в котором в алгоритме планирования был использован подход выбора узлов для задач, предложенный в данной работе.

В таблице 1 приведены результаты выполнения задач из тестового набора при различных вариантах запуска. Указаны минимальное, среднее и максимальное время расчета задачи для всех трех вариантов и показатели интенсивности работы с оперативной памятью на вычислительных узлах кластера для вариантов, в которых задачи выполнялись в конкурентном режиме.

Таблица 1. Результаты расчета тестового набора моделей при различных вариантах запуска

Вариант запуска	(1)	(2)	(3)
Минимальное время, с	111	116	124
Среднее время, с	580	695	677
Максимальное время, с	891	1094	979
Средняя интенсивность работы с памятью на узле, МБ/с		39188	42718
Стандартное отклонение по узлам, МБ/с		8505	3699

В результате использования предложенного подхода распределения многопоточных программ с учетом конкуренции за доступ к подсистеме памяти, было снижено время выполнения тестового набора задач. Кроме того, при использовании предложенного алгоритма нагрузка на оперативную память на узлах кластера становится более сбалансированной, и, следовательно, повышается средняя интенсивность работы с памятью на задействованных узлах кластерной системы.

6. Заключение

В данной работе построена модель памяти двухпроцессорного вычислительного узла с NUMA архитектурой. Данная модель позволяет оценить временные задержки, возникающие на устройствах подсистемы памяти при выполнении многопоточной программы, интенсивно использующей оперативную память. Кроме того, приведенная модель используется для оценки замедления времени выполнения приложения в конкурентном режиме, то есть при работе более одной программы на вычислительном узле. Это оценка используется при выборе узла для запуска задачи на кластерной системе.

В результате тестирования описанного алгоритма распределения задач было выявлено, что при его использовании возможно снижение времени выполнения многопоточных программ в конкурентном режиме, при этом нагрузка на оперативную память узлов становится более сбалансированной. Все эксперименты проведены на примере гидродинамического симулятора нефтяных и газовых месторождений.

Данный подход распределения многопоточных программ по узлам кластерной системы внедрен в алгоритм планирования кластерного планировщика задач CSched, разработанного и используемого в Уфимском государственном авиационном техническом университете.

Список литературы

[1] S. Blagodurov, S. Zhuravlev, M. Dashti, A. Fedorova. *A Case for NUMA-Aware Contention Management on Multicore Processors* // USENIX Annual Technical Conference. — Portland, OR, USA, 2011, June. ↑2

[2] Л. Клейнрок. Теория массового обслуживания: Пер. с англ. И. И. Грушко / ред. В. И. Нейман. Москва: Машиностроение, 1979. — 432 с. ↑3, 4

[3] User and Reference Guide for the Intel®C++ Compiler 14.0. ↑5

[4] С. А. Айвазян, В. С. Мхитарян. Прикладная статистика и основы эконометрики. Москва: ЮНИТИ, 1998. — 1010 с. ↑6

Об авторах:

Рауль Аликович Ахметшин

Инженер кафедры Высокопроизводительных вычислительных технологий и систем

e-mail: raul.akhmetshin@gmail.com

Артур Владимирович Юлдашев

Старший преподаватель кафедры Высокопроизводительных вычислительных технологий и систем

e-mail: arthur@mail.rb.ru

Образец ссылки на эту публикацию:

Р. А. Ахметшин, А. В. Юлдашев. *Распределение многопоточных программ на многопроцессорных системах с неоднородным доступом к памяти* // Программные системы: теория и приложения: электрон. научн. журн. 2014. Т. ??, № ?, с. ??–??.

URL: http://psta.psiras.ru/read/

Raul Akhmetshin, Arthur Yuldashev. *Multithreaded Programs Distribution on Multiprocessor NUMA Systems.*

Abstract. In this paper the model of state-of-the-art NUMA multiprocessor memory system is proposed. This model allows to estimate time degradation of multithreaded applications running on such systems. Also we proposed job distributing algorithm for cluster systems which consider contention for memory system on modern multiprocessor nodes. This algorithm designed with proposed memory system model. The possibility of reducing the application runtime on cluster systems using suggested job distributing algorithm is experimentally shown. (*in Russian*).

Key Words and Phrases: Multiprocessor system, Cluster systems, High-performance computing, NUMA.