ОПЫТ РАСПАРАЛЛЕЛИВАНИЯ АЛГОРИТМА ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ ЗАДАЧИ СИНТЕЗА СТРАТЕГИЙ ОБСЛУЖИВАНИЯ ПОТОКА ОБЪЕКТОВ

А.С. Куимова

Волжская государственная академия водного транспорта, Нижний Новгород
E-mail: anastasia.kuimova@gmail.com

Рассматриваются математическая модель и экстремальная задача обслуживания объектов конечного детерминированного потока стационарным процессором. На содержательном уровне рассматриваемая модель описывает технологию обслуживания грузовых судов на терминале с учетом времени их прибытия в порт. Требования экономической эффективности эксплуатации подобных объектов и оперативности диспетчерского управления накладывают жесткие ограничения на допустимую длительность формирования управляющих воздействий. В силу сказанного актуальной является проблема разработки достаточно быстрых алгоритмов синтеза оптимальных стратегий обслуживания потока объектов, одно из перспективных решений которой связано с распараллеливанием вычислений. В рамках технологии OpenMP [2] для многопроцессорных вычислительных систем с общей памятью разрабатываются и исследуются возможные схемы распараллеливания решающих алгоритмов динамического программирования [1].

Математическая модель обслуживания и постановка оптимизационной задачи

Считается заданным конечный детерминированный поток объектов $O_n = \{o_1, o_2, \ldots, o_n\}$, которые подлежат однофазному обслуживанию на стационарном процессоре P. Для каждого объекта o_i определены следующие целочисленные характеристики: t_i – момент поступления в очередь на обслуживание ($0 \le t_1 \le t_i \le \ldots \le t_n$), τ_i – норма длительности обслуживания, a_i – штраф за единицу времени простоя в ожидании обслуживания. Если обслуживание объекта o_i начинается в момент $t_i^* \ge t_i$, то величина штрафа по этому объекту равна $a_i(t_i^* - t_i)$. Полагаем, что процессор P готов к обслуживанию потока объектов в момент времени $t = 0$. Обслуживание объекта o_i ($i = \overline{1,n}$) может быть начато процессором в любой момент времени t ($t \ge t_i$) и осуществляется без прерываний; необслуженный объект не может покинуть очередь к процессору; одновременное обслуживание двух и более объектов и непроизводительные простои процессора запрещены.

Стратегия обслуживания S представляет собой произвольную перестановку $S = \{i_1, i_2, \ldots, i_n\}$ совокупности индексов $N = \{1, 2, \ldots, n\}$. Считаем, что при реализации стратегии S объект с индексом i_k будет обслуживаться k-м по очереди ($k = \overline{1,n}$).

Обозначим через $t^*(i_k, S)$ момент начала обслуживания объекта с индексом i_k при реализации стратегии S. Очевидно, что $t^*(i_1, S) = t_{i_1}$; $t^*(i_k, S) = \max(t^*(i_{k-1}, S) + \tau_{i_{k-1}}, t_{i_k})$, $k = \overline{2,n}$. Тогда совокупный штраф за простои объектов потока O_n в ожидании обслуживания согласно стратегии S будет: $K_1(S) = \sum_{k=1}^{n} a_{i_k}(t^*(i_k, S) - t_{i_k})$.

С целью повышения эффективности диспетчерского управления потоком объектов решается задача синтеза стратегии обслуживания S, минимизирующая значение суммарного штрафа $K_1(S)$:

$$\min_S (K_1(S)). \qquad (1)$$

Последовательный алгоритм синтеза оптимальной стратегии

Построение алгоритма синтеза оптимальной стратегии выполним, опираясь на формализм динамического программирования. Для этого введем следующие обозначения: $F(t)$ – подмножество индексов объектов, поступающих в систему обслуживания в момент времени t; $D(t, \Delta)$ – совокупность индексов объектов, прибывающих в систему обслуживания на интервале времени $[t + 1, t + \Delta]$, $\Delta \geq 1$. Очевидно, что $D(t,\Delta) = \bigcup_{i=1}^{\Delta} F(t+i)$. Считаем, что пара значений (t, Q) определяет текущее состояние системы в момент t принятия решения, где Q – множество индексов объектов потока O_n, ожидающих обслуживания в этот момент времени.

Пусть $E(t, Q)$ – минимальная величина суммарного штрафа по объектам множества Q и всем объектам, поступающим в систему обслуживания позднее момента времени t для определяемой состоянием (t, Q) ситуации. Как очевидно, $E(0, F(0))$ – оптимальное значение критерия в исходной задаче (1).

Для любого $\theta \geq 0$ и $Q = \{\alpha\}$, где α – произвольный индекс объекта из потока O_n, имеет место соотношение

$$E(t_n + \theta, \{\alpha\}) = a_\alpha \cdot (t_n + \theta - t_\alpha). \qquad (2)$$

Если в состоянии (t, Q) в качестве очередного обслуживаемого объекта выбран объект с индексом α, то очередным моментом принятия решения будет $t + \tau_\alpha$, а выбор индекса следующего объекта для обслуживания будет осуществляться из множества $(Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha)$. Поэтому

$$E(t,Q) = \min_{\alpha \in Q}[(a_\alpha(t - t_\alpha) + E(t + \tau_\alpha, (Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha)))]. \qquad (3)$$

Соотношения (2), (3) суть решающие задачу (1) рекуррентные соотношения. Их вычислительная реализация по классической схеме [3] – алгоритм ALG – связана со значительными временными затратами. Ниже рассматриваются варианты распараллеливания вычислений, направленные на сокращение продолжительности синтеза стратегии S.

Параллельные реализации алгоритмов синтеза стратегии обслуживания

Очевидно, что при решении задачи (1) с использованием рекуррентных соотношений (2), (3) перебор объектов множества Q может быть выполнен параллельно на различных узлах многопроцессорной вычислительной системы. При выполнении данной работы были разработаны и экспериментально исследованы две схемы распараллеливания.

Первая схема подразумевает распределение всех подзадач, полученных на определенной глубине рекурсии R, по вычислительным узлам; соответствующий этой схеме алгоритм будем именовать как PALG1. Реализация этого алгоритма включает в себя последовательное выполнение следующих трех этапов.

На первом этапе формируются все подзадачи, порожденные алгоритмом ALG на заданной глубине рекурсии R; на втором этапе полученные подзадачи динамически распределяются по доступным вычислительным узлам; на третьем этапе выполняется решение порожденных подзадач, из множества полученных на всех узлах многопро-

цессорной системы решений редуцируется оптимальная оценка и соответствующая ей стратегия обслуживания.

Первый этап алгоритма PALG1 можно представить следующей схемой.

Шаг 1. Сформировать начальное состояние системы обслуживания и задать начальное текущее решение.

Шаг 2. Выполнить вычисления алгоритмом ALG на одном узле до достижения заданного количества шагов рекурсии *R*.

Шаг 3. Запомнить в глобальном хранилище полученную подзадачу $E(t, Q)$, последовательность индексов уже обслуженных объектов и соответствующую оценку.

Шаг 4. Если на заданном уровне рекурсии *R* список необслуженных объектов *Q* не пуст, рассмотреть следующий объект и перейти к шагу 3, иначе к шагу 5.

Шаг 5. Осуществить рекурсивный выход из функции. Если список необслуженных объектов на предыдущих уровнях рекурсии не пуст, перейти к шагу 2, иначе к шагу 6.

Шаг 6. Перейти к выполнению второго этапа алгоритма.

На втором этапе алгоритма PALG1 сохраненные в глобальном хранилище подзадачи в цикле динамически распределяются по имеющимся *m* вычислительным потокам. Для этого первые *m* подзадач выбираются из хранилища и раздаются потокам для вычисления, дальнейшая выборка подзадач осуществляется в момент решения одним из потоков своей подзадачи.

Третий этап алгоритма PALG1 включает следующие шаги.

Шаг 1. Суммировать оценку, полученную в результате решения каждой подзадачи, с соответствующей оценкой, сохраненной в глобальном хранилище.

Шаг 2. Дополнить полученную в результате решения подзадачи последовательность индексов обслуженных объектов соответствующей последовательностью индексов, сохраненной в глобальном хранилище, формируя стратегию обслуживания.

Шаг 3. Сравнить полученное в результате шагов 1 и 2 допустимое решение исходной задачи с текущим решением, сохраненным в глобальном хранилище. Если полученное решение дает меньшее значение критерия, то заменяем им текущее.

Шаг 4. Шаги 1-3 выполняются по завершении решения одной подзадачи любым из вычислительных узлов.

В результате выполнения трех этапов алгоритма PALG1 в глобальном хранилище будет находиться итоговое решение исходной задачи в виде минимальной оценки и соответствующей стратегии.

Второй подход основан на распараллеливании циклов по объектам множества *Q* на определенной глубине рекурсии по ходу выполнения последовательного алгоритма. Соответствующий алгоритм будем именовать как PALG2, представим его в виде следующей схемы.

Шаг 1. Сформировать начальное состояние системы обслуживания и задать начальное текущее решение.

Шаг 2. Выполнить вычисления алгоритмом ALG на одном вычислительном узле до достижения заданного количества шагов рекурсии *R*.

Шаг 3. Распределить полученные подзадачи динамически по доступным вычислительным узлам. При этом если подзадача одна, то распараллеливание не выполняется, если подзадач от 2 до 3, то в решении участвуют два вычислительных узла, во всех остальных случаях подзадачи распределяются по четырем узлам.

Шаг 4. Выполнить решение подзадач по схеме третьего этапа алгоритма PALG1. Сравнить полученное решение с текущим, если оно лучше, то заменить им текущее.

Шаг 5. Осуществить рекурсивный выход из функции. Если список необслуженных объектов на предыдущих уровнях рекурсии не пуст, перейти к шагу 2, иначе закончить вычисления.

В результате работы алгоритма мы получим оптимальное значение критерия и соответствующую стратегию обслуживания в текущем решении задачи.

Результаты вычислительных экспериментов

С целью исследования и оценки времени, затрачиваемого на поиск решения одного и того же набора задач последовательным алгоритмом ALG и параллельными алгоритмами PALG1 и PALG2 был выполнен ряд экспериментов. Вычисления выполнялись на компьютере с двухъядерным процессором Intel Core i5, 2,4 ГГц (ОЗУ 4 Гб). Программа разрабатывалась на языке C с использованием компилятора GCC 4.6. Результаты экспериментов приведены в таблице 1 и на рис. 1. Длительности T отработки алгоритмов синтеза стратегии S указаны в секундах. Ускорение U оценивалось отношением средних длительностей решения задач последовательным алгоритмом ALG и параллельными алгоритмами PALG1 (PALG2) при соответствующих значениях n величины размерности потока O_n. Параллельный алгоритм PALG2 запускался с использованием двух и четырех вычислительных узлов процессора, тогда как в алгоритме PALG1 количество используемых вычислительных узлов определяется динамически.

Таблица 1. Результаты вычислительных экспериментов

n	ALG	PALG1		PALG2, 2 узла		PALG2, 4 узла	
		T, с	U	T, с	U	T, с	U
8	0,001	0,002	0,500	0,002	0,500	0,002	0,500
9	0,004	0,007	0,571	0,007	0,571	0,006	0,667
10	0,019	0,032	0,594	0,03	0,633	0,028	0,679
11	0,173	0,129	1,341	0,112	1,545	0,102	1,696
12	0,919	0,574	1,601	0,568	1,618	0,378	2,431
13	8,964	4,682	1,915	3,956	2,266	3,789	2,366
14	34,826	18,86	1,847	16,136	2,158	13,046	2,669
15	303,826	137,239	2,214	145,456	2,089	121,231	2,506

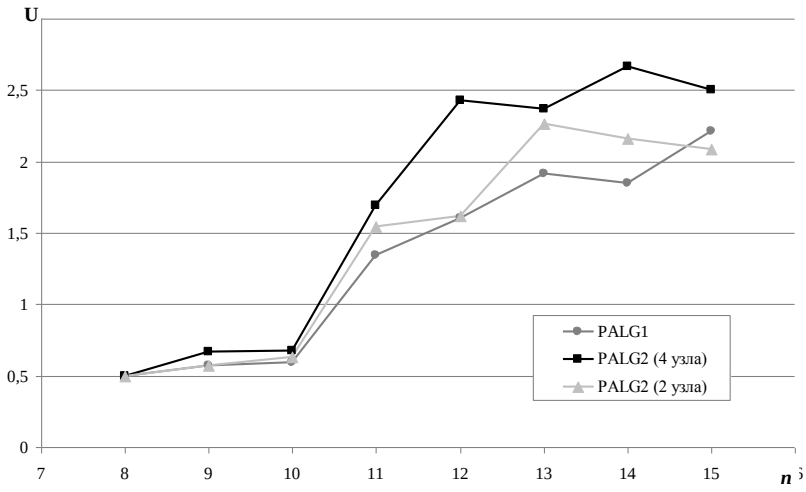

Рис. 1. Зависимость ускорения U от размерности потока O_n

Предложенные в работе схемы распараллеливания алгоритма динамического программирования показали достаточно высокую эффективность при решении задач синтеза стратегий обслуживания потока объектов. Результаты экспериментов позволяют рекомендовать разработанные алгоритмы для реализации в специализированных си-

стемах поддержки оперативного планирования транспортно-технологических процессов, в которых числом объектов больше 13.

Литература

1. Беллман Р., Дрейфус С. Прикладные задачи динамического программирования. – М.: Наука, 1965. – 457 с.
2. Левин М.П. Параллельное программирование с использованием OpenMP. – М.: БИНОМ. Лаборатория знаний, 2008. –120 с.
3. Коган Д.И., Федосенко Ю.С. Задача диспетчеризации: анализ вычислительной сложности и полиномиально разрешимые подклассы // Дискретная математика. 1996. Т. 8. Вып. 3. С. 135-147.
4. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. – 608 с.
5. Гергель В.П. Теория и практика параллельных вычислений. – М.: 2007. – 424 с.