УДК 681.324.001.57

Садыхов Р.Х., Отвагин А.В

АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ О НАЗНАЧЕНИИ В МУЛЬТИПРОЦЕССОРНОЙ СИСТЕМЕ НА ОСНОВЕ МОДЕЛИ ВИРТУАЛЬНОЙ СЕТИ

ВВЕДЕНИЕ

Современные высокопроизводительные вычислительные системы широко используют различные методы параллельной обработки. При этом поддержку параллельных вычислений осуществляют различные специализированные библиотеки, обеспечивающие интерфейс обмена информацией и синхронизации параллельных процессов. В качестве примера можно привести широко распространенный в мире стандартизованный интерфейс передачи сообщений (Message Passing Interface - MPI)[1]. Данный подход использует модульное построение параллельной программы, при котором параллельный алгоритм представлен в виде графа потенциально параллельных модулей, выполняемых на разных устройствах обработки. При этом возникает задача оптимального распределения модулей по процессорам с точки зрения скорости выполнения, минимума задержек по передаче информации, надежности и устойчивости вычислений.

1. ПОСТАНОВКА ЗАДАЧИ

В общем случае применительно к многопроцессорным вычислительным системам задача о назначении формулируется следующим образом. Пусть существует множество процессоров P, каждый из которых имеет производительность p_i $\{i = 1,\ldots,n\}$. Процессоры соединены между собой линиями связи пропускной способности $c_{i,j}$. Данную систему можно представить в виде направленного ациклического графа. Пусть также существует параллельный алгоритм, представленный в виде подобного графа, в котором определено множество программных модулей M. Каждый модуль имеет вычислительную сложность m_k, определяющую время его вычисления. Модули обмениваются между собой информацией, при этом между двумя модулями существует поток информации объемом $f_{k,l}$.

Результатом работы оптимизирующего алгоритма является такое распределение модулей по процессорам, при котором достигается максимальная скорость вычислений и передачи информации. При этом требуется минимизировать функцию вида:

$$\min z = \sum \frac{m_k}{p_i} + \sum \frac{f_{k,l}}{c_{i,j}}. \qquad (1)$$

2. ОБЩИЕ АЛГОРИТМЫ РЕШЕНИЯ

Проблема назначения программных модулей на процессоры системы с оптимальным распределением нагрузки может быть достаточно легко решена переборными методами, если учитывать только процессы вычисления. Однако наличие информационной связи между модулями вносит определенную сложность в решение данной задачи и приводит к необходимости разработки алгоритмов, учитывающих этот факт.

В [2] решение задачи о назначении реализовано методами кластеризации (разделения) графа на подграфы, каждый из которых содержит задачи, назначенные на отдельный процессор. Известен ряд алгоритмов кластеризации [3-5], однако они обеспечивают оптимальные решения на определенном подмножестве задач, но не обеспечивают решения в общем случае. Ниже рассмотрены основные идеи этой группы алгоритмов.

Алгоритмы кластеризации основаны на так называемом «обнулении дуг». При этом дуги обмена информацией между задачами одного процессора считаются исключенными, а вес или стоимость коммуникации этих дуг принимается равным 0. При определении суммарной оценки эти дуги не влияют на ее величину. На одном шаге алгоритм может обнулять в зависимости от реализации определенное число дуг.

Каждый алгоритм использует при кластеризации определенные эвристики. Эти эвристики определяют дуги, которые должны быть «обнулены». Обычно эвристики задаются в виде функции стоимости определенного вида, позволяющей ее вычисление за полиномиальное время. При этом часто рассматриваются некоторые специальные случаи графов задач, которые дают возможность легко сформулировать функцию оценки.

Алгоритмы кластеризации также предполагают использование техники «поиска без возврата»(non-backtracking). При этом на конечную цель алгоритма накладывается условие, препятствующее рассмотрению конфигураций, являющихся худшими по отношению к текущему решению или множеству решений. Таким образом, обеспечивается непрерывная сходимость результатов применения алгоритма к оптимуму. Обычно в качестве такого ограничения рассматривается монотонное уменьшение времени выполнения параллельного алгоритма.

Различные постановки целей оптимизации накладывают свои особенности на результат кластеризации. Все алгоритмы, указанные в [3-5], дают различные результаты в общем случае, и эффективно применимы только на ограниченном множестве задач. Между тем, в отечественной литературе решение задачи о назначении рассматривалось в более широкой постановке, с учетом различных целей оптимизации. В частности, предложенный в [6] алгоритм нахождения минимального разреза потока в сети учитывает одновременно время выполнения задачи на совокупности процессоров и время обменов информацией между задачами. Алгоритм использует для решения модель модифицированного графа межмодульных связей. В этом графе кроме дуг, определяющих передачу информации, присутствуют дуги, определяющие время выполнения отдельной задачи на каждом из процессоров. Разрез графа однозначно определяет распределение модулей по процессорам, а вес разреза является общей ценой выполнения программы при данном распределении модулей по процессорам.

3. МОДЕЛЬ ВИРТУАЛЬНОЙ СЕТИ

В данной работе алгоритм поиска решения основан на модели виртуальной нейронной сети, основные концепции которой определены в [7]. В этой модели совокупность элементов графа разделяется на кластеры. Каждый элемент кластера тесно связан с другими элементами этого кластера, т.е. цена связей внутри кластера значительно выше, чем цена внешних связей этого кластера. Связь называется внутренней для кластера, если вершины ее начала и окончания принадле-

Садыхов Рауф Хосровович. *Д.т.н., профессор, зав. каф. ЭВМ, БГУИР.*
Отвагин Алексей Владимирович. *Аспирант, Институт технической кибернетики НАНБ.*

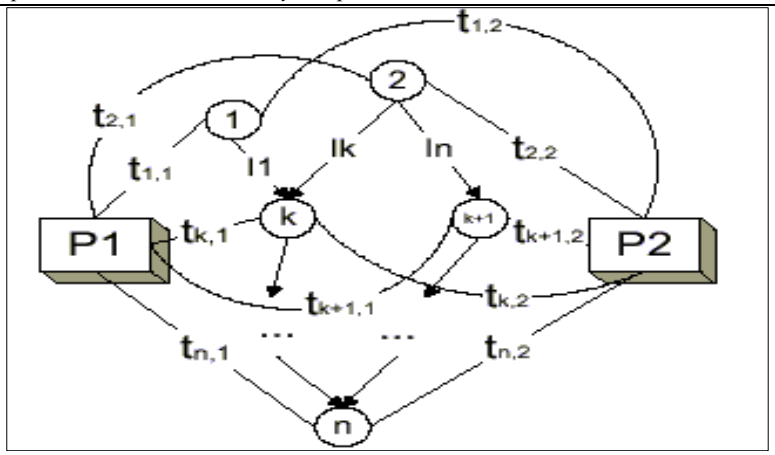

Рисунок 1 – Расширенный граф для двух процессоров.

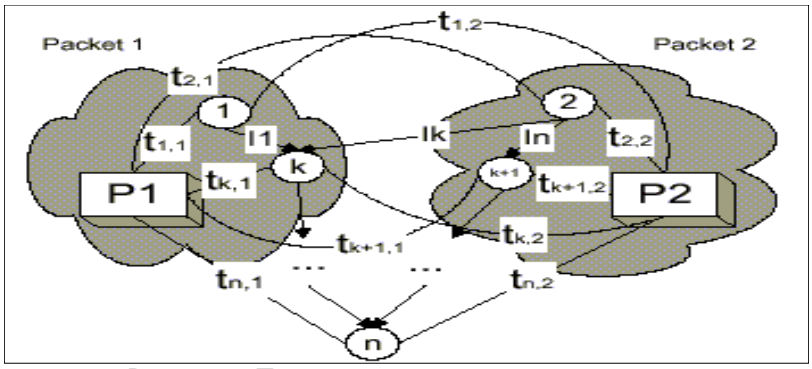

Рисунок 2 – Пример кластеризации виртуальной сети.

жат данному кластеру. Данное правило соответствует общему для всех упомянутых алгоритмов принципу обнуления наиболее «дорогих» связей. Модель виртуальной сети легко масштабируется на произвольное число процессоров и задач. Данная модель предполагает самоорганизацию, т.е. выравнивание оценок для отдельных кластеров и стремление их к некоторой средней величине.

Способность модели к самоорганизации в процессе ее определения или изменения позволяет при правильном выборе критериев оценки получить разбиение графа задач на процессоры с близкой к оптимальной ценой выполнения на вычислительной системе.

Процедура формирования кластеров основана на определении для них параметра, называемого прочностью. Прочность представляет собой отношение сумм весов связей между элементами кластера и весов связей, внешних по отношению к кластеру. В процессе самоорганизации виртуальной сети она стремится сформировать кластеры таким образом, чтобы их прочности были максимальны и по возможности одного порядка. Рассмотрим некоторый кластер C, для которого справедливы следующие выражения:

$$W_C^{int} = \sum_{i,j \in C} c_{i,j} \; ; \qquad (2)$$

$$W_C^{ext} = \sum_{i \in C,\, j \notin C} c_{i,j} \; ; \qquad (3)$$

$$S_C = W_C^{int} \; / \; W_C^{ext} \; , \qquad (4)$$

где $c_{i,j}$ – вес дуги между элементами i и j, W_C^{int} – сумма весов дуг внутри кластера C; W_C^{ext} – сумма весов дуг между кластером C и другими кластерами, а S_C – прочность кластера C.

Поскольку в процессе кластеризации виртуальной сети непосредственную роль играет понятие прочности кластера, было бы разумным строить функцию оценки отдельной модели именно на ней. Чтобы учесть примерное равенство прочностей кластеров, возьмем в качестве общей оценки сумму взаимных произведений частных прочностей для каждого кластера

$$S = \sum_{i=1}^{N} \sum_{j=i}^{N} S_i * S_j \; . \qquad (5)$$

Процесс самоорганизации виртуальной сети предполагает рассмотрение множества возможных конфигураций. Таким образом, требуется решить NP-сложную задачу выбора одного из вариантов, возможно, не самого лучшего. Наилучшим методом, используемым для этого, представляется метод, основанный на генетических алгоритмах [8,9].

Генетические алгоритмы основаны на механизмах естественной селекции и генетики. Их преимущество состоит в простоте, надежности и внутреннем параллелизме. Они менее подвержены ограничениям, накладываемым на пространство поиска и выполняют параллельный поиск в нескольких направлениях на одной итерации. Генетические алгоритмы наиболее подходят для оценки множества возможных комбинаций, представляющих решение. Для определения задачи в терминах генетических алгоритмов используется термин «популяция» Каждая популяция состоит из «хромосом», представляющих собой один из вариантов условного описания исследуемой модели. Описание условно, поскольку методика генетических алгоритмов применима к достаточно широкому кругу задач.

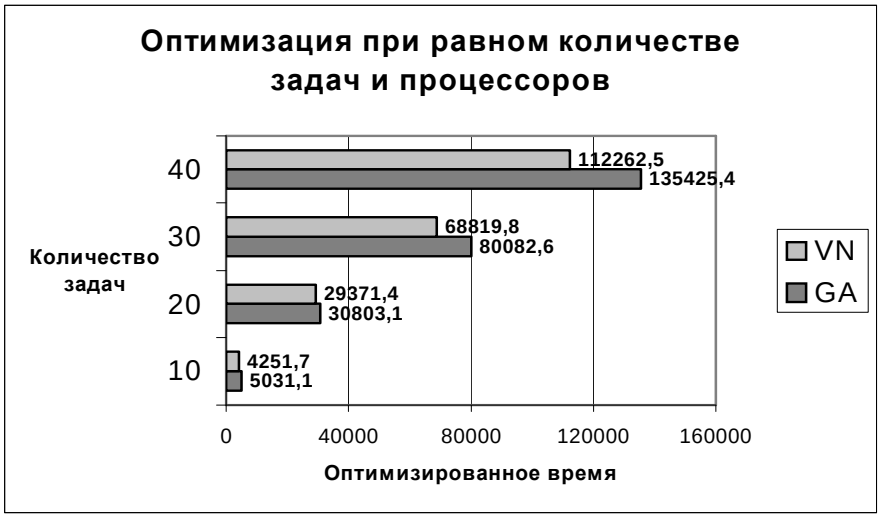

Рисунок 3 – Результаты первой группы экспериментов.

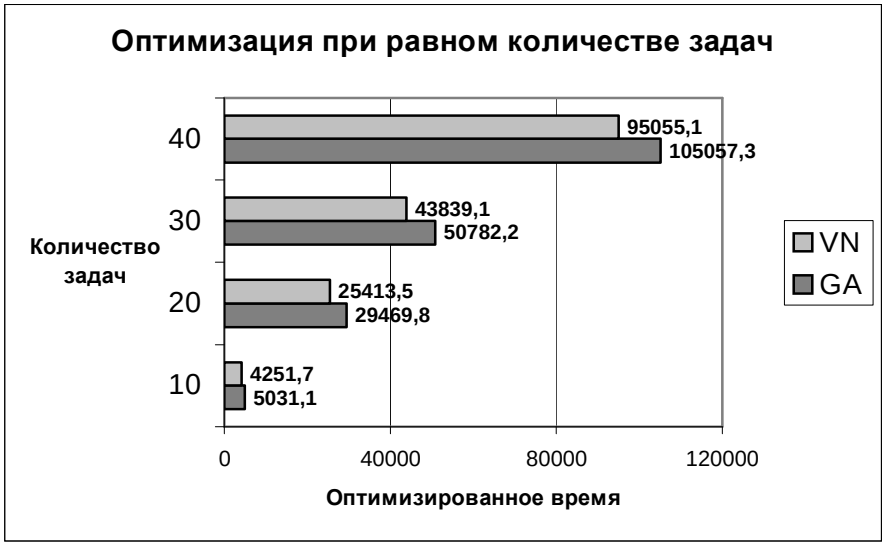

Рисунок 4 – Результаты второй группы экспериментов.

Поскольку модель виртуальной нейронной сети предполагает обучение в процессе самоорганизации, то техника генетических алгоритмов будет далее расширена методами теории нейронных сетей.

4. АЛГОРИТМ ОБУЧЕНИЯ ВИРТУАЛЬНОЙ СЕТИ

Для представления графа задач в виде виртуальной сети его необходимо расширить с учетом информации о производительности процессоров, на которых этот граф будет выполнен. Для этого необходимо построить граф, в который будут добавлены вершины, соответствующие процессорам и дуги, соединяющие каждую вершину, соответствующую задаче и вершину, соответствующую процессору (рисунок 1).

Каждой дуге будет присвоен вес, соответствующий времени выполнения задачи на определенном процессоре. Вес вычисляется по формуле (1) как первое слагаемое.

В процессе кластеризации виртуальной сети определенные вершины образуют кластер совместно с вершиной одного из процессоров. Таким образом, конечное число кластеров не превышает количество процессоров. Вершины, попавшие в один кластер с процессором, определяют подмножество задач, назначенных на данный процессор. Пример кластеризации приведен на рисунке 2.

Первоначально все дуги виртуальной сети получают либо нулевые веса, либо вес дуги – случайное небольшое число. После этого строится начальная кластеризация. При использовании генетических алгоритмов в качестве хромосомы выступает целочисленный вектор размерности N, где N – число задач алгоритма управления. Каждая позиция вектора принимает случайное значение от 0 до M, где M – количество процессоров вычислительной системы. Все позиции, имеющие одно значение, представляют собой кластер соответствующих задач на одном процессоре. Хромосома, таким образом, описывает вариант разделения виртуальной сети на отдельные кластеры.

После этого проводится оценка каждого варианта кластеризации по формуле (5). Эта оценка используется для поиска новых вариантов кластеризации в генетическом алгоритме.

При обучении модели виртуальной сети используется принцип Хебба [10] - изменение веса связей, способствующих получению лучшего решения. Связи, попавшие в любой кластер, изменяют вес в соответствии с:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha T_i , \qquad (6)$$

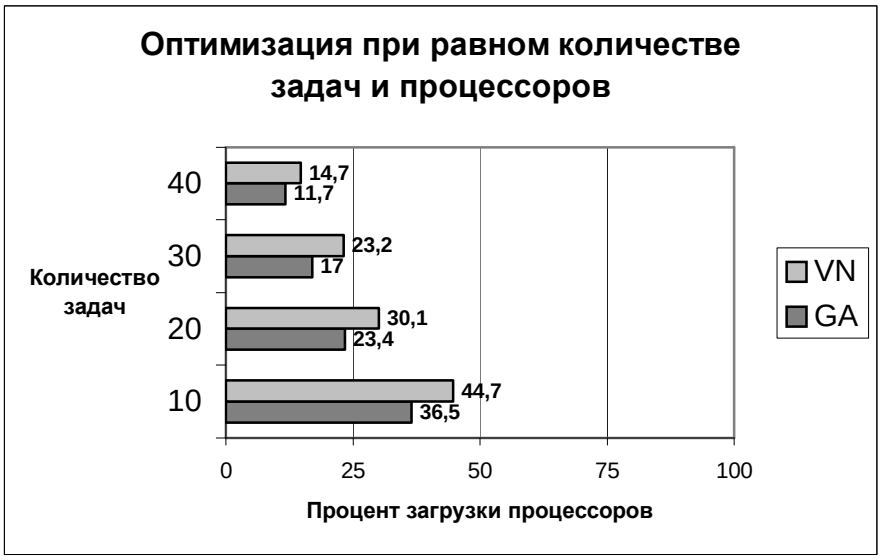

Рисунок 5 – Первая группа экспериментов по оценке средней загрузки.

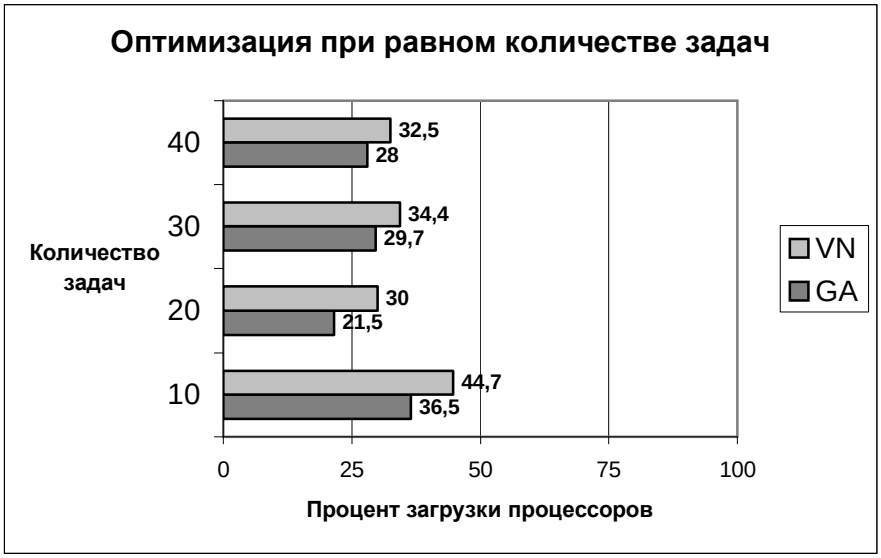

Рисунок 6 – Вторая группа экспериментов по оценке средней загрузки.

где α - параметр обучения, T_i – величина, обратная оценке наилучшего варианта. В данном случае минимум T_i означает меньшее увеличение веса дуги и, как следствие, меньшее «удаление» ее от определенного процессора. При дальнейшей кластеризации шансы попасть в кластер у дуг с меньшими весами гораздо выше. Если оценка была удачна, полученная информация о конфигурации кластеров накапливается в виртуальной сети и используется в дальнейшем обучении. Таким образом, в процессе обучения происходит непрерывное изменение конфигурации виртуальной сети для достижения оптимального решения.

5. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

Для оценки преимуществ модели виртуальной сети были проведены сравнительные эксперименты. Для этого была создана программная модель решения задачи о назначении, реализующая оптимизационную процедуру методом генетических алгоритмов и с помощью модели виртуальной сети. Кроме этого, модель содержит процедуры определения временных и качественных показателей эффективности работы алгоритмов.

Первая группа экспериментов состояла из задачи распределения равного количества задач по такому же количеству процессоров. Для каждого алгоритма проведена серия из 40 экспериментов для каждой группы задач. Результаты представлены на рисунке 3. Здесь GA – результаты генетического алгоритма, VN – результаты виртуальной сети.

Вторая группа экспериментов состояла из задачи распределения количества задач, кратного количеству процессоров. Была также проведена серия из 40 экспериментов для каждой группы задач. Число процессоров было фиксировано и равно 10. Результаты приведены на рисунке 4.

Кроме того, при проведении экспериментов учитывалось распределение вычислительной нагрузки по процессорам системы. Для этого определялось время вычислений на каждом процессоре, и максимальное из времен принималось за 100%. После этого загрузка L определялась по формуле

$$L = \left(\sum_i \frac{L_i}{L_{max}} \right) * \frac{1}{N_{used}}, \qquad (7)$$

где L_i – нагрузка на процессоре i, L_{max} – максимальная нагрузка, N_{used} – используемое реально число процессоров.

Условия экспериментов по оценке средней загрузки были такими же, как и в случае оценки оптимизации по времени выполнения. Результаты оценки средней загрузки процессоров показывают, что алгоритм виртуальной сети обеспечивает более рациональное использование процессоров и равномерность их загрузки, нежели метод поиска решения с помощью генетических алгоритмов. Соответствующие результаты приведены на рисунке 5 и рисунке 6.

ЗАКЛЮЧЕНИЕ

Проведенные эксперименты показали эффективность использования модели виртуальной сети для решения задачи о назначении модулей параллельной программы на процессоры многопроцессорной вычислительной системы. С увеличением количества задач рост производительности алгоритма виртуальной сети заметно увеличивается. Кроме того, алгоритм виртуальной сети наряду с улучшением времени выполнения параллельной программы увеличивает равномерность загрузки процессоров и способствует более эффективному их использованию для вычислений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. И Евсеев. MPI – программный инструмент для параллельных вычислений. http://www.csa.ru/~il/Parallel/myMPI/intro.htm.
2. A Comparison of Clustering Heuristics for Scheduling DAGs on Multiprocessors. A.Gerasoulis and T. Yang. Department of Computer Science, Rutgers University, New Brunswick, NJ 08903.
3. Kim S.J., and Browne J.C. A General Approach to Mapping of Parallel Computation upon Multiprocessor Architectures, International Conference on Parallel Processing, vol. 3, 1988, pp. 1-8
4. Sarkar V. Partitioning and Scheduling Parallel Programs for Execution on Multiprocessors, The MIT Press, 1989.
5. Yang T., Gerasoulis A. A Fast Static Algorithm for DAGs on an Unbounded Number of Processors, Proc. of Supercomputing'91, IEEE, 1991, pp. 633-642.
6. Трахтенгерц Э.А. Программное обеспечение параллельных процессов. М.: Наука, 1987.
7. Yufik Y. M., Sheridan T. B. Virtual Networks: New framework for operator modeling and interface optimization in complex supervisory control systems // A Rev. Control, vol. 20, pp. 179-195.
8. Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs. Second, Extended Edition. Springer-Verlag, 1994, 340 pp.
9. Курейчик В.М. Генетические алгоритмы: обзор и состояние.//«Новости искусственного интеллекта», №1, 1997. С. 14-63.
10. В.А. Головко. Нейроинтеллект: Теория и применение. Кн.1:Организация и обучение нейронных сетей с прямыми и обратными связями. Брест, 1999.

УДК 681.3.001.57

Садыхов Р.Х, Радишевский В.А, Отвагин А.В.

АЛГОРИТМ СТАТИЧЕСКОЙ МАРШРУТИЗАЦИИ СООБЩЕНИЙ В УВК РЕАЛЬНОГО ВРЕМЕНИ НА БАЗЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

(Примечание: Работа выполнена при поддержке гранта БРФФИ Т00-050)

ВВЕДЕНИЕ

В современных управляющих вычислительных комплексах (УВК) и высокопроизводительных вычислительных системах наряду с проблемой оптимального распределения задач по устройствам обработки существует проблема взаимодействия и синхронизации процессов. Эти проблемы наиболее остро стоят при использовании режимов реального времени для функционирования управляющих вычислительных комплексов (УВК РВ). Современные вычислительные системы часто используют для обеспечения взаимодействия процессов специализированные средства операционных систем, либо библиотеки обмена сообщениями. В качестве примера можно привести широко распространенный в мире стандартизованный интерфейс передачи сообщений (Message Passing Interface - MPI)[1]. Данный интерфейс предполагает использование идентификаторов для обмена сообщениями между процессами. Для гарантированного обмена при старте программы создается специальная коммуникационная область, объединяющая все процессоры в единое коммуникационное пространство. Данный подход применяется, как правило, в сильно локализованных высокопроизводительных системах. Однако, применение его в достаточно распределенных системах, таких, как промышленные SCADA-системы (Supervisory Control And Data Acquisition Systems) неоправданно, поскольку управление полным коммуникационным пространством достаточно трудоемкая процедура.

Оптимизация распределения задач по процессорам системы снимает часть проблем взаимодействия и синхронизации процессов. Однако оставшиеся межпроцессные связи, ставшие «межпроцессорными», нуждаются в продуманной и оптимальной организации для обеспечения требуемых режимов функционирования. Отсюда и вытекает задача оптимальной маршрутизации в УВК и особенно УВК РВ.

1. ПОСТАНОВКА ЗАДАЧИ

Пусть существует множество процессоров P, каждый из которых имеет производительность по обработке собственных или транзитных сообщений p_i $\{i = 1,\ldots,n\}$. Процессоры соединены между собой линиями связи пропускной способности $c_{i,j}$. Данную систему можно представить в виде направленного ациклического графа G_p. Пусть также существует параллельный алгоритм, в котором определено множество программных модулей M. Модули обмениваются между собой информацией, при этом между двумя модулями существует поток информации объемом $f_{k,l}$. Все модули некоторым образом распределены по узлам коммуникационной сети G_p, в которой и происходит обмен информацией. Информация каждой взаимодействующей пары модулей передается

Радишевский Валерий Анатольевич. *К.т.н., доцент. каф. ЭВМ, БГУИР.*