ПРОГРАММИРОВАНИЕ, 2020, № 6, с. 55–66

ПАРАЛЛЕЛЬНОЕ И РАСПРЕДЕЛЕННОЕ ПРОГРАММИРОВАНИЕ

УДК 004.724.4

ОТОБРАЖЕНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ, ИСПОЛЬЗУЮЩИЕ ТЕХНОЛОГИЮ RapidIO

© 2020 г. Н. И. Вьюкова[a,*], В. А. Галатенко[a,], А. Н. Павлов[a,***], С. В. Самборский[a,****]**

[a] Федеральное государственное учреждение "Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук", Москва, Нахимовский проспект, д. 36, к. 1, 117218 Россия

**E-mail: qniva@yandex.ru*

***E-mail: galat@niisi.ras.ru*

****E-mail: antony@niisi.msk.ru*

*****E-mail: sambor@niisi.ras.ru*

Поступила в редакцию 18.05.2020 г.
После доработки 15.06.2020 г.
Принята к публикации 17.06.2020 г.

Статья посвящена вопросам отображения параллельных вычислений на распределенные системы с коммутационной средой RapidIO. Распределенная система представляет собой множество вычислительных узлов с заданными величинами производительности и узлов-коммутаторов, связанных сетевыми соединениями с заданными величинами пропускной способности. Под параллельным вычислением понимается множество процессов, взаимодействующих посредством передачи потоков данных, где для каждого потока требуется определенная пропускная способность сетевого маршрута, а для каждого процесса – определенная производительность. Требуется назначить вычислительные узлы для процессов и определить маршруты для потоков данных, а также построить таблицы маршрутизации для коммутаторов так, чтобы обеспечить заданные пропускные способности для потоков данных и заданные величины производительности для процессов. Данная задача в общем случае является NP-полной. В работе предложен подход, основанный на методах целочисленного линейного программирования (ЦЛП), позволяющий корректно отобразить параллельное вычисление на распределенную систему и оптимизировать отображение по ряду характеристик.

DOI: 10.31857/S0132347420060084

1. ВВЕДЕНИЕ

Процессоры семейства Комдив64, разработанные в ФГУ ФНЦ НИИСИ РАН, предназначены для широкого спектра высокопроизводительных вычислительных приложений, в том числе, приложений для управления различными техническими устройствами в реальном времени.

Для эффективной работы подобных приложений может требоваться большой объем вычислительных ресурсов. Одним из общепринятых способов реализации высокопроизводительных вычислений в настоящее время является применение распределенных вычислительных систем. Важными преимуществами этого подхода являются масштабируемость и гибкость, поскольку вычислительный комплекс может быть спроектирован в соответствии с потребностями приложения.

В ФГУ ФНЦ НИИСИ РАН разработаны процессорные и мезонинные модули на базе микропроцессоров КОМДИВ64-РИО и КОМДИВ128-РИО, а также коммутаторы и другие аппаратные средства для построения многопроцессорных комплексов, связанных посредством коммуникационной сети RapidIO [1].

RapidIO – это высокопроизводительный интерфейс передачи данных для соединения микросхем на одной печатной плате, а также для соединения между собой нескольких печатных плат. Данный интерфейс был разработан для применения во встраиваемых системах. RapidIO оптимизирован для энергоэффективных связей процессор-процессор с минимальными задержками в устойчивых к ошибкам системах, узлы которых удалены друг от друга на расстояние не более ки-

лометра. Согласно [2], по таким характеристикам как пропускная способность, задержки при передаче пакетов, надежность, масштабируемость, RapidIO превосходит другие коммуникационные средства.

В [3] представлена технология разработки систем цифровой обработки сигналов для многопроцессорных комплексов на базе процессоров архитектуры Комдив и коммуникационной среды RapidIO (Комдив-RapidIO). В [4] рассмотрен опыт реализации алгоритма геометрического многосеточного метода из пакета NAS Parallel Benchmarks на распределенном комплексе Комдив-RapidIO. Поддержка коммуникационной сети RapidIO на уровне операционной системы реального времени (ОСРВ) Багет представлена в [5, 6]. На прикладном уровне создание программного обеспечения для комплексов Комдив-RapidIO обеспечивается библиотекой параллельной обработки сигналов (БПОС) [7]. Реализован также ряд инструментальных средств для конфигурирования и загрузки параллельных программ на распределенные комплексы Комдив-RapidIO [8], а также для интерактивной ручной оптимизации привязки параллельно выполняемых процессов к вычислительным узлам целевого многопроцессорного комплекса [9].

Тем не менее, вопрос оптимального отображения параллельного вычисления на распределенную вычислительную систему остается нерешенным. Процессы взаимодействуют между собой путем передачи сообщений. Для эффективного выполнения параллельного вычисления в целом необходимо, чтобы сетевые маршруты обеспечивали определенный темп передачи данных между взаимодействующими процессами. Этого можно добиваться как путем варьирования назначения вычислительных узлов для процессов, так и путем изменения схемы маршрутизации в сети RapidIO. При этом вычислительный узел, назначаемый для каждого процесса, должен обладать определенными свойствами, такими как уровень производительности или наличие некоторого сопроцессора.

Для небольшого числа процессов оптимальное (или удовлетворительное) отображение и подходящая схема маршрутизации могут быть подобраны вручную экспериментальным путем. Но когда число процессов исчисляется десятками, такой экспериментальный подбор оказывается весьма трудоемким и не гарантирует оптимального результата.

Статья посвящена вопросам точного решения задачи об отображении заданного параллельного вычисления на распределенную вычислительную систему с сетью произвольной топологии и статической маршрутизацией сетевой среды. Данная задача является NP-полной, и в настоящей работе для ее решения предложен подход, использующий методы целочисленного линейного программирования (ЦЛП).

Последующая часть статьи построена по следующему плану. В разделе 2 обсуждаются существующие работы в данной области. В разделе 3 рассматриваются формальные модели параллельного вычисления и распределенной вычислительной системы на базе коммуникационной среды RapidIO. В разделе 4 представлены формулировки двух ЦЛП-задач – задачи нахождения схемы маршрутизации в условиях, когда процессам уже назначены узлы распределенной системы, и более сложной задачи, включающей назначение вычислительных узлов для процессов и нахождение схемы маршрутизации. В разделе 5 приведены примеры решения этих двух ЦЛП-задач. В заключение анализируются результаты проведенных исследований и рассматриваются направления дальнейших исследований.

2. ДРУГИЕ РАБОТЫ В ЭТОЙ ОБЛАСТИ

В сфере высокопроизводительных вычислений требования к вычислительным мощностям постоянно возрастают. В настоящее время практически единственный способ удовлетворить эти растущие требования заключается в использовании параллельных вычислительных конфигураций, в частности, распределенных многопроцессорных систем. В связи с этим большое значение приобретает развитие методов, позволяющих автоматически распределить параллельно выполняемые процессы по узлам компьютерной сети с соблюдением тех или иных требований. Исследования в данной области ведутся уже на протяжении не одного десятилетия. Однако их актуальность остается высокой в силу постоянного развития архитектур распределенных систем и сетевых технологий, а также из-за разнообразия требований, которые могут предъявляться к искомым отображениям.

Многие варианты задачи отображения параллельных вычислений на распределенные системы являются NP-полными, поэтому для ее решения преимущественно используются эвристические подходы. Обзор подобных подходов можно найти в [10]. В связи с настоящим исследованием наибольший интерес представляют работы, посвященные точным методам решения данной задачи с использованием таких формализмов как ЦЛП, смешанное целочисленное линейное программи-

рование (Mixed Integer Linear Programming, MILP), задача выполнимости формул в теориях (Satisfiability Modulo Theories, SMT). Применение точных методов оправдано, если отображение параллельной задачи на многопроцессорную конфигурацию определяется в процессе ее разработки и отладки производительности. Точные методы могут применяться для оценки эффективности эвристических алгоритмов, как в [11], [12]. Они также могут применяться в сочетании с эвристическими подходами, как в [13].

Формулировки задач отображения параллельных вычислений на распределенную вычислительную систему могут существенно отличаться, во-первых, из-за различий архитектур используемых вычислительных систем и коммуникационных моделей, во-вторых, из-за различий в критериях допустимости и оптимальности искомых отображений.

Значительное число работ в данной области посвящено многопроцессорным системам на кристалле (Multiprocessor system-on-chip, MPSoC) с сетью, имеющей структуру 2D-решетки. Например, в [14] рассматривается выполнение параллельных вычислений на системе MPSoC и ставится задача найти отображение множества процессов на множество процессоров и определить приоритеты коммуникаций между процессами так, чтобы минимизировать общее время выполнения. В работах [15, 16] рассматриваются вопросы минимизации коммуникационных издержек между взаимодействующими процессами, выполняющимися в многопроцессорных конфигурациях, за счет дублирования процессов на нескольких вычислительных узлах.

Целый ряд работ посвящен актуальным темам отображения параллельных вычислений на многопроцессорные вычислительные системы с оптимизацией энергопотребления [13, 17, 18] или тепловыделения [11].

В ряде случаев при поиске отображений параллельных вычислений на распределенную систему приходится рассматривать многокритериальную оптимизацию. Например, в [19] отображения параллельных вычислений на узлы MPSoC оптимизируется как по сбалансированности загрузки узлов, так и по коммуникационным издержкам. Обычно в таких случаях применяют целевую функцию, являющуюся линейной комбинацией двух (или более) параметров. В [19] предлагается подход, позволяющий находить “неулучшаемые” решения из множества Парето.

В большинстве работ формулировки ЦЛП или MILP-задач включают спецификацию расписания для параллельных вычислений, с тем чтобы удовлетворить заданные временные ограничения для систем реального времени, как в [11, 13], либо минимизировать общее время выполнения, см. [14, 16].

В данной работе временные характеристики выполнения параллельных вычислений не рассматриваются. Основной ее вклад состоит в том, что, помимо отображения процессов на вычислительные узлы распределенной системы, в результате решения ЦЛП-задачи вычисляется также схема маршрутизации, которая гарантирует требуемую пропускную способность маршрутов между вычислительными узлами, на которых выполняются взаимодействующие процессы. В других работах либо подразумевается детерминированный алгоритм маршрутизации, например, XY-маршрутизация в системах MPSoC, (см. [11, 16]), либо предполагается, что таблица маршрутизации предопределена, как в [20].

В [20] представлена постановка задачи, наиболее близкая к нашей. В этой работе используется формализм SMT для поиска отображения параллельно выполняемых процессов на вычислительные узлы сети произвольной топологии с целью минимизации общего сетевого трафика и обеспечения сбалансированной загрузки вычислительных узлов. Как и в нашей работе, в [20] учитываются ограничения по пропускной способности каждого сетевого соединения: суммарная потребность всех маршрутов, использующих некоторое соединение, не должна превышать пропускной способности этого соединения. Однако в [20], в отличие от нашей работы, таблица маршрутизации считается предопределенной, и варьируется только назначение вычислительных узлов для процессов.

3. ФОРМАЛЬНЫЕ МОДЕЛИ

3.1. Модель распределенной вычислительной системы

Коммуникационная среда RapidIO представляет собой сеть с коммутацией пакетов, состоящую из узлов, соединенных физическими каналами связи [1]. Структура распределенных систем на базе аппаратуры линейки Комдив под управлением ОСРВ Багет, подробно описана в [5]. Формальная модель такой распределенной системы представлена в [21]. В данной работе, как и в [6], мы будем рассматривать упрощенную модель, включающую узлы двух типов:

– вычислительный узел – оконечное устройство, имеющее числовой идентификатор и, как правило, только один порт;

– коммутатор – устройство, предназначенное для маршрутизации, которое всегда имеет несколько портов.

Коммутатор работает под управлением таблицы маршрутизации, которая записывается во время инициализации коммуникационной среды RapidIO. Есть два типа коммутаторов. Коммутатор первого типа имеет одну общую таблицу маршрутизации для всех портов, в коммутаторе второго типа имеется индивидуальная таблица для каждого порта.

В коммутаторах первого типа выходной порт, с которого будет отправлен полученный пакет данных, однозначно определяется указанным в этом пакете идентификатором вычислительного узла – получателя. Это означает, что если маршруты, по которым передаются данные одному получателю, пересекаются на некотором коммутаторе, то далее они совпадают.

В коммутаторах второго типа выходной порт, с которого будет отправлен полученный пакет данных, определяется идентификатором получателя и номером порта, на который этот пакет пришел. Таким образом, пакеты к одному получателю могут уйти с коммутатора по разным соединениям, если они поступили в него по разным соединениям.

Последовательность соединений, через которые проходит пакет от отправителя до получателя, называется *маршрутом*. Множество маршрутов для всех пар отправителей и получателей, между которыми происходит передача данных, называется *схемой маршрутизации*.

Представим теперь формальную модель распределенной системы, которая будет использована в формулировках ЦЛП-задач в разделе 4.

Распределенная система представляется в виде ориентированного размеченного (нагруженного) графа

$$G = (H, C_1, C_2, L, b, perf),$$

где H – множество вычислительных узлов, C_1 – множество коммутаторов первого типа, C_2 – множество коммутаторов второго типа. Обозначим через *Node* множество вычислительных узлов и коммутаторов: $Node = H \cup C_1 \cup C_2$. Тогда $L \subseteq Node \times Node$ – множество дуг, соответствующих соединениям между узлами распределенной системы. Отображения b: $L \to \mathbb{N}$ и *perf*: $H \to \mathbb{N}$ задают, соответственно, пропускную способность соединений и производительность вычислительных узлов в некоторых условных единицах.

На практике в среде RapidIO соединения двунаправленные, с одинаковой пропускной способностью в обе стороны. Поэтому граф G можно было бы определить как неориентированный. Тем не менее мы рассматриваем его как ориентированный, поскольку в формулировках ЦЛП-задач соединения (n, n') и (n', n) определяются независимо друг от друга.

3.2. Модель параллельного вычисления

Параллельное вычисление мы будем представлять в виде ориентированного размеченного графа

$$PG = (P, S, v, req)$$

где P – множество процессов, реализующих параллельное вычисление, $S \subseteq P \times P$ – множество дуг, связывающих взаимодействующие процессы: $s = (p, p') \in S$, если процесс p отправляет сообщения процессу p'. Допускаются как ациклические графы, так и графы с циклами. Дуги $s \in S$ мы далее будем называть потоками данных. Отображения v: $S \to \mathbb{N}$ и *req*: $P \to \mathbb{N}$ определяют, соответственно, требуемую для каждого потока данных пропускную способность сетевого маршрута и требуемую для каждого процесса производительность вычислительного узла.

4. ФОРМУЛИРОВКИ ЦЛП-ЗАДАЧ

В этом разделе мы рассмотрим две формулировки ЦЛП-задач, связанных с отображением параллельного вычисления на распределенную вычислительную систему.

В первой задаче предполагается, что множество процессов P параллельного вычисления $PG = (P, S, v, req)$ уже отображено на множество вычислительных узлов H распределенной вычислительной системы $G = (H, C_1, C_2, L, b, perf)$. Требуется определить схему маршрутизации, обеспечивающую заданную пропускную способность для потоков данных между процессами, а также вычислить таблицы маршрутизации для коммутаторов.

Формулировка второй ЦЛП-задачи включает подбор вычислительных узлов из H для процессов из P, а также нахождение схемы и таблиц маршрутизации.

В качестве инструмента для описания и решения ЦЛП-задач применялся пакет GLPK [22].

4.1. Задача маршрутизации

Пусть заданы параллельное вычисление $PG = (P, S, v, req)$ и распределенная вычислительная система $G = (H, C_1, C_2, L, b, perf)$. Первая ЦЛП-задача, которую мы рассмотрим, это задача вычис-

ления схемы маршрутизации и таблиц маршрутизации в условиях, когда процессы параллельного вычисления уже назначены на вычислительные узлы этой системы. В этой задаче отображения *perf* и *req*, связанные с производительностью, не рассматриваются.

Пусть определено отображение m: $P \to H$. Тем самым однозначно определяется множество $S' \subseteq \subseteq H \times H$ потоков данных между вычислительными узлами: $s' = (h_1, h_2) \in S'$, если существует поток $s = (p_1, p_2) \in S$, такой что $m(p_1) = h_1$, $m(p_2) = h_2$. Будем называть узел h_1 отправителем, а h_2 получателем потока данных s'. Требуемая пропускная способность для потока s', $v'(s')$, полагается равной сумме $v(s)$ по всем $s = (p_1, p_2) \in S$, таким что $m(p_1) = h_1$, $m(p_2) = h_2$.

Константы, переменные, отображения и подмножества, используемые в формулировках 1-й и 2-й ЦЛП-задач.

Константный массив LN определяет, как связаны соединения $l \in L$ и вершины $n \in Node$ графа G, LN: $L \times Node \to \{-1, 0, 1\}$. Пусть $l = (n_1, n_2)$ – сетевое соединение, направленное от узла n_1 к узлу n_2.

$$LN[l,n] = \begin{cases} -1 & iff \quad n = n_1 \\ 1 & iff \quad n = n_2 \\ 0 & iff \quad n \neq n_1, \quad n \neq n_2 \end{cases}$$

Аналогично введем константный массив SN, определяющий связь между потоками данных и узлами вычислительной системы: SN: $S' \times Node \to$ $\to \{-1, 0, 1\}$. Пусть $s = (n_1, n_2)$ – поток данных от узла n_1 к узлу n_2.

$$SN[s,n] = \begin{cases} -1 & iff \quad n = n_1 \\ 1 & iff \quad n = n_2 \\ 0 & iff \quad n \neq n_1, \quad n \neq n_2 \end{cases}$$

Значение 0 показывает, что узел n не является ни начальным, ни конечным для потока s (хотя может являться промежуточным узлом в маршруте для потока s). Этот массив используется только в формулировке первой ЦЛП-задачи.

Введем также обозначения для следующих подмножеств соединений: $Lout_1$ – множество соединений, исходящих из коммутаторов первого типа; $Lout_2$ – множество соединений, исходящих из коммутаторов второго типа; Lin_2 – множество соединений, входящих в коммутаторы второго типа.

Отношение следования, *next*, на множестве соединений используется в формулировках ограничений, обеспечивающих связность маршрутов: *next*(l, l'), если соединение l входит в коммутатор, из которого исходит l'.

Основная переменная, описывающая искомые маршруты для потоков данных из S', это переменная R, которая имеет тип массива бинарных значений: R: $L \times S' \to \{0, 1\}$. Для $l \in L$, $s \in S'$ значение $R[l, s] = 1$, если соединение l используется в маршруте для потока данных s; в противном случае $R[l, s] = 0$.

Массив переменных TC_1 определяет таблицы маршрутизации для коммутаторов первого типа, TC_1: $Lout_1 \times H \to \{0, 1\}$. $TC_1[l, h] = 1$, если соединение $l \in Lout_1$ используется для передачи сообщений, адресованных вычислительному узлу $h \in H$.

Массив переменных TC_2 определяет таблицы маршрутизации для коммутаторов второго типа, TC_2: $Lin_2 \times Lout_2 \times H \to \{0, 1\}$. Для $h \in H$, $l \in Lin_2$, $l' \in Lout_2$, $TC_2[l, l', h] = 1$, если выходное соединение l' используется для дальнейшей пересылки сообщений узлу h, пришедших по входному соединению l. Это имеет смысл, если l и l' связаны с одним коммутатором; переменные для l и l', не связанных с одним коммутатором, не создаются.

В определениях переменных TC_1 и TC_2 сам коммутатор, для которого заполняется таблица маршрутизации, явно не фигурирует. Но для каждой переменной этих массивов коммутатор однозначно определяется по соединению (l для TC_1, l или l' для TC_2).

Маршруты и таблицы коммутации должны удовлетворять ряду ограничений. Это ограничения, связанные со спецификой коммутационной среды RapidIO, ограничения, обеспечивающие заданную пропускную способность для потоков данных между процессами, ограничения, исключающие маршруты с петлями.

Ограничение на маршруты для потоков данных. Следующее ограничение указывает, что каждый маршрут возникает в узле-отправителе соответствующего потока данных и заканчивается в узле-получателе, а в остальные узлы из *Node* входит столько же раз, сколько выходит (не более одного, если маршрут не имеет петель):

$$\forall s \in S', \quad \forall n \in Node$$
$$\sum_{l \in L} LN[l,n] \cdot R[l,s] = SN[s,n] \qquad (1)$$

Ограничения по пропускным способностям. Сумма требуемых пропускных способностей потоков данных, чьи маршруты проходят через некоторое соединение, не должна превышать пропускную способность соединения:

$$\forall l \in L \quad \sum_{s \in S'} v'(s) \cdot R[l,s] \le b(l) \tag{2}$$

Ограничения для вычислительных узлов. Вычислительный узел не может быть транзитным, то есть ни в одном маршруте не может быть использовано более одного соединения, связанного с таким узлом:

$$\forall h \in H, \quad \forall s \in S' \quad \sum_{l \in L: LN[l,h] \ne 0} R[l,s] \le 1 \tag{3}$$

Ограничения на вход и выход из коммутаторов. Маршрут не может использовать два или более соединений, входящих в один и тот же коммутатор, а также два или более соединений, исходящих из одного и того же коммутатора:

$$\forall c \in C_1 \cup C_2, \quad \forall s \in S'$$

$$\sum_{l \in L: LN[l,c]=1} R[l,s] \le 1 \tag{4}$$

$$\forall c \in C_1 \cup C_2, \quad \forall s \in S'$$

$$\sum_{l \in L: LN[l,c]=-1} R[l,s] \le 1 \tag{5}$$

Эти ограничения исключают маршруты с петлями, проходящие через один и тот же коммутатор второго типа более одного раза. Наличие таких маршрутов не является ошибкой для коммутаторов типа 2, но мы исключаем их из соображений эффективности передачи данных.

Ограничения для таблиц маршрутизации коммутаторов первого типа. Следующее ограничение отражает специфику коммутаторов первого типа:

$$\forall h \in H, \quad \forall c \in C_1$$

$$\sum_{l \in Lout_1: LN[l,c]=-1} TC_1[l,h] \le 1 \tag{6}$$

Оно указывает, что для каждого коммутатора $c \in C_1$ и для каждого вычислительного узла h, не более одного соединения, исходящего из c, предназначено для пересылки сообщений узлу h.

Следующее ограничение указывает, что соединение l, исходящее из коммутатора первого типа, может быть использовано в маршруте для потока передачи данных s, ведущего в вычислительный узел h, только если l "предназначено" для пересылок в узел h.

$$\forall l \in Lout_1, \quad \forall s = (h', h) \in S'$$

$$R[l,s] \le TC_1[l,h] \tag{7}$$

Ограничения для таблиц маршрутизации коммутаторов второго типа. Следующее ограничение описывает специфику коммутаторов второго типа:

$$\forall h \in H, \quad \forall c \in C_2, \quad \forall l \in Lin_2 : LN[l,c] = 1$$

$$\sum_{l' \in Lout_2: LN[l',c]=-1} TC_2[l,l',h] \le 1 \tag{8}$$

Для каждого вычислительного узла h, для каждого коммутатора c второго типа и для каждого соединения l, входного для c, не более одного выходного соединения из этого же коммутатора может быть предназначено для пересылки сообщений, пришедших по входному соединению l и адресованных узлу h.

Наконец, определим ограничение на использование соединений из Lin_2, $Lout_2$ в маршрутах для потоков данных.

$$\forall s = (h', h) \in S' \quad \forall l \in Lin_2, \quad \forall l' \in Lout_2 : next(l,l')$$

$$R[l,s] + R[l',s] \le TC_2[l,l',h] + 1 \tag{9}$$

Переменные R и TC_2 принимают значения 0, 1, то есть их можно рассматривать как булевы переменные. Поэтому неравенство (9) эквивалентно логическому высказыванию

$$R[l,s] \;\&\; R[l',s] \Rightarrow TC_2[l,l',h]$$

Таким образом, два соединения l, l', из которых первое входит в коммутатор, а второе исходит из него, могут быть использованы в маршруте для потока данных, только если l' "предназначено" для пересылок получателю этого потока, пришедших по соединению l.

Целевая функция. Представленные выше ограничения гарантируют подбор маршрутов, обеспечивающих заданные пропускные способности для всех потоков данных. С помощью целевой функции можно обеспечить оптимизацию решения по различным параметрам. В данной реализации минимизируется линейная комбинация следующих переменных: максимальная длина маршрута ($R_{\max}$), суммарная длина маршрутов (R_{total}), суммарное число записей во всех таблицах маршрутизации (TC_{total}).

Для переменной $R_{\max}$ определены следующие ограничения:

$$\forall s \in S' \quad \sum_{l \in L} R[l,s] \le R_{\max} \tag{10}$$

Переменная R_{total} определяется как сумма элементов массива R, а переменная TC_{total} – как сум-

ма элементов TC_1 и TC_2. Минимизируемая целевая функция имеет следующий вид:

$$1000 \cdot R_{\max} + 10 \cdot R_{\text{total}} + TC_{\text{total}} \qquad (11)$$

Отметим, что ограничения (1)–(9) не запрещают несвязные маршруты с циклами, отдельными от основного маршрута. Такие маршруты исключаются только оптимизацией по R_{total} в целевой функции.

Несложно модифицировать целевую функцию, чтобы оптимизировать решение и по некоторым другим параметрам, которые обсуждаются в заключении.

4.2. Задача подбора вычислительных узлов и схемы маршрутизации

Пусть заданы параллельное вычисление $PG = (P, S, v, req)$ и распределенная вычислительная система $G = (H, C_1, C_2, L, b, perf)$. Формулировка второй ЦЛП-задачи отличается от представленной выше тем, что включает подбор вычислительных узлов из H для процессов из P. Допускается отображение нескольких процессов на один узел, имеющий достаточную производительность. Но отображение одного процесса на несколько узлов невозможно. Если возникает такая необходимость, например, когда требуемая для процесса производительность превышает производительность имеющихся вычислительных узлов, такой процесс должен быть заранее распараллелен на несколько процессов.

В этой формулировке будут использованы константы, переменные и другие обозначения, введенные в п. 4.1, а также несколько дополнительных констант и одна переменная.

Константы и переменные, используемые в формулировке 2-й ЦЛП-задачи

В предыдущей формулировке мы использовали константный массив SN, определяющий связь между потоками данных и узлами вычислительной системы. В данной формулировке мы введем вместо этого константный массив: SP: $S \times P \to \{-1, 0, 1\}$, описывающий связь между потоками данных и процессами параллельного вычисления. Пусть $s = (p_1, p_2)$ – поток данных от процесса p_1 к процессу p_2.

$$SP[s, p] = \begin{cases} -1 & iff \quad p = p_1 \\ 1 & iff \quad p = p_2 \\ 0 & iff \quad p \neq p_1, \quad p \neq p_2 \end{cases}$$

Искомое отображение множества процессов P на множество вычислительных узлов H представим в виде переменной M, имеющей тип массива бинарных значений: M: $H \times P \to \{0, 1\}$. Для $h \in H$, $p \in P$ значение $M[h, p] = 1$, если процессу p назначен вычислительный узел h.

Ограничения на отображение M. С отображением M связано два очевидных ограничения. Первое – каждому процессу должен быть назначен в точности один вычислительный узел:

$$\forall p \in P \quad \sum_{h \in H} M[h, p] = 1 \qquad (12)$$

Второе – сумма потребностей в производительности процессов, назначенных на один вычислительный узел, не должна превышать производительности этого узла:

$$\forall h \in H$$
$$\sum_{p \in P} req(p) \cdot M[h, p] \leq perf(h) \qquad (13)$$

Ограничение на маршруты для потоков данных. В п. 4.1 мы ввели ограничение (1), указывающее, что каждый маршрут возникает в узле-отправителе и заканчивается в узле-получателе, а в остальные узлы из *Node* входит столько же раз, сколько выходит. Теперь должно быть учтено отображение процессов на вычислительные узлы. Для вычислительных узлов это ограничение принимает следующий вид:

$$\forall s \in S, \quad \forall h \in H$$
$$\sum_{l \in L} LN[l, h] \cdot R[l, s] = \sum_{p \in P} SP[s, p] \cdot M[h, p] \qquad (14)$$

В коммутаторе поток заведомо не начинается и не заканчивается; для коммутаторов это ограничение выглядит следующим образом:

$$\forall s \in S, \quad \forall c \in C_1 \cup C_2$$
$$\sum_{l \in L} LN[l, c] \cdot R[l, s] = 0 \qquad (15)$$

Ограничение (2) по пропускной способности соединений, ограничение (3), запрещающее использовать вычислительные узлы как транзитные, и ограничения (4), (5), запрещающие многократное прохождение маршрута через коммутатор, остаются без существенных изменений. Отличие лишь в том, что вместо множества потоков данных между вычислительными узлами S' используется множество потоков данных между процессами S.

Ограничения для коммутаторов первого и второго типа. Ограничения (6) и (8), отражающие специфику коммутаторов первого и второго типов, остаются без изменений, поскольку они сформулированы только в терминах распределенной системы.

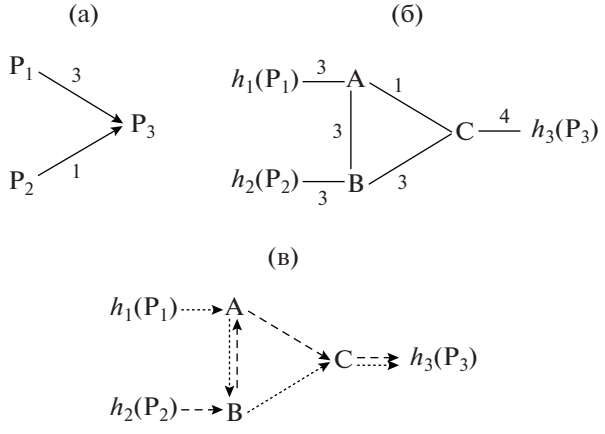

Рис. 1. Пример 1, решение первой ЦЛП-задачи: а) параллельное вычисление; б) распределенная система с заданным отображением процессов на вычислительные узлы; в) решение, маршруты для потоков данных $P_1 \to P_3$ и $P_2 \to P_3$.

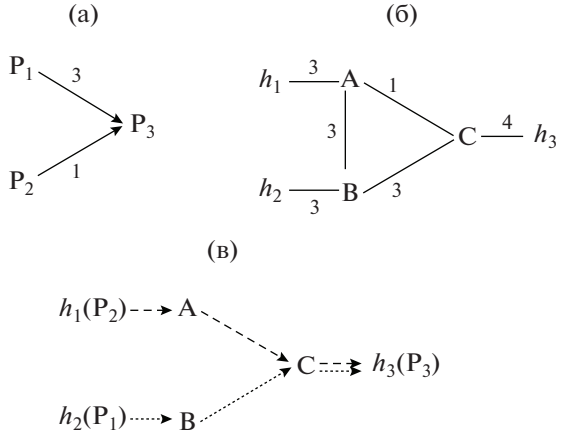

Рис. 2. Пример 1, решение второй ЦЛП-задачи: а) параллельное вычисление; б) распределенная система; в) решение, отображение процессов на вычислительные узлы и маршруты для потоков $P_1 \to P_3$ и $P_2 \to P_3$.

Небольшие отличия возникают в ограничениях (7) и (9) в связи с тем, что в первой формулировке мы рассматривали потоки данных между вычислительными узлами из H, а в данной формулировке потоки данных соединяют процессы из P. Теперь соответствующие ограничения описывают, что в маршруте для потока данных процессу-получателю p можно использовать только выходные соединения, предназначенные таблицей маршрутизации для узла, в который назначен процесс p. Таким образом, ограничение (7) приобретает вид:

$$\forall l \in Lout_1, \quad \forall s = (p', p) \in S, \quad \forall h \in H$$

$$M[h, p] + R[l, s] \le TC_1[l, h] + 1 \qquad (16)$$

А ограничение (9) заменяется на:

$$\forall s = (p', p) \in S,$$
$$\forall l \in Lin_2, \quad \forall l' \in Lout_2 : next[l, l'], \quad \forall h \in H$$

$$M[h, p] + R[l, s] + R[l', s] \le TC_2[l, l', h] + 2 \qquad (17)$$

Переменные, связанные с целевой функцией, ограничение (10), а также целевая функция (11) определяются так же, как в первой задаче.

5. ПРИМЕРЫ

В этом разделе мы рассмотрим примеры решения ЦЛП-задач, представленных выше. Примеры демонстрируют работоспособность предложенных моделей, а также эффект от использования целевой функции. Простой пример, представленный на рис. 1, иллюстрирует специфику двух типов коммутаторов и различие между формулировками двух ЦЛП-задач.

На рис. 1а) показан граф параллельного вычисления из трех процессов P_1, P_2, P_3 с потоками данных $P_1 \to P_3$ и $P_2 \to P_3$ с требованиями по пропускной способности 3 и 1 соответственно.

На рис. 1б) показан граф распределенной системы с тремя вычислительными узлами h_1, h_2, h_3 и отображенными на них процессами P_1, P_2, P_3 соответственно. Распределенная система включает также три коммутатора: A и B – второго типа, C – первого типа. Сплошными линиями показаны пары разнонаправленных соединений между узлами, а цифры показывают пропускные способности соединений. На рис. 1в) показано решение первой ЦЛП-задачи: маршруты, соответствующие потокам данных $P_1 \to P_3$ (пунктирные линии) и $P_2 \to P_3$ (штриховые линии).

Можно видеть, что коммутаторы A и B направляют сообщения адресату h_3 с разных портов в зависимости от того, с какого узла пришло сообщение, поэтому здесь существенно, что оба они относятся ко второму типу. Легко видеть, что это решение единственное, поэтому для случая, когда хотя бы один из коммутаторов A, B относится к первому типу, решения не существует.

Хотя суммарная потребность в пропускной способности для двух потоков данных, проходящих между коммутаторами A и B, равна 4, а на рисунке для (A, B) указана пропускная способность 3, полученное решение корректно, поскольку эти потоки используют два разнонаправленных соединения: (A, B) и (B, A), каждое с пропускной способностью 3.

Рассмотрим теперь решение 2-й ЦЛП-задачи для аналогичного примера, который изображен на рис. 2. Здесь считается, что все вычислительные узлы имеют производительность 1, и потреб-

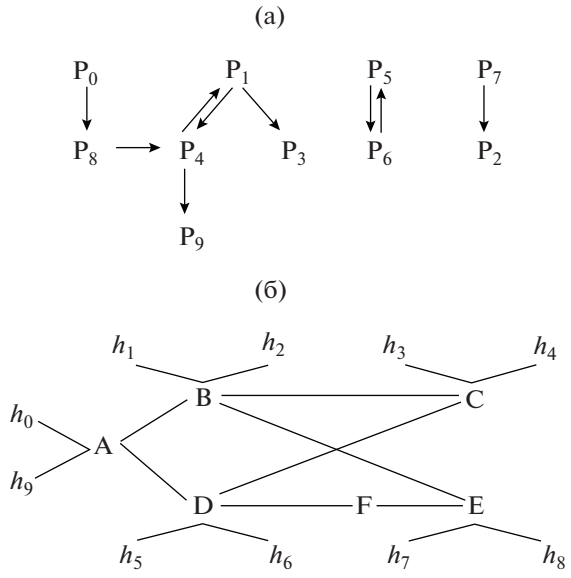

Рис. 3. Пример 2. а) параллельное вычисление; б) распределенная система.

ности всех процессов в производительности также равны 1. На рис. 2а) показан граф параллельного вычисления. На рис. 2б) изображен граф распределенной системы, но без предопределенного отображения процессов на вычислительные узлы. На рис. 2в) показано решение второй ЦЛП-задачи – отображение процессов на вычислительные узлы и маршруты, соответствующие потокам данных $P_1 \to P_3$ и $P_2 \to P_3$.

Назначение вычислительных узлов для процессов, найденное в результате решения второй ЦЛП-задачи, позволило проложить более короткие маршруты для потоков данных $P_1 \to P_3$ и $P_2 \to P_3$, чем в предыдущем случае, без встречных пересылок по соединению (A, B). В данном случае все коммутаторы могут быть первого типа.

Рассмотрим теперь пример более сложных входных данных, представленный на рис. 3. Параллельное вычисление (рис. 3а) включает 10 процессов с требованиями по производительности 10 единиц. Для всех потоков данных, показанных на рисунке, требуется пропускная способность 40 единиц. Распределенная система (рис. 3b) содержит 10 вычислительных узлов h_0, ..., h_9 производительностью по 10 единиц, и коммутаторы A, B, C, D, E, F. Пропускная способность соединений, связывающих вычислительные узлы с коммутаторами – 100 единиц, соединения между коммутаторами имеют пропускную способность 50 единиц.

Первая ЦЛП-задача при назначении процессов P_0, ..., P_9 на узлы h_0, ..., h_9 соответственно решения не имеет. В данном случае причину уста-

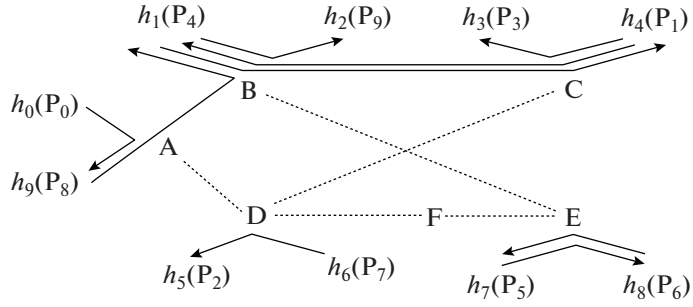

Рис. 4. Пример 2, решение второй ЦЛП-задачи.

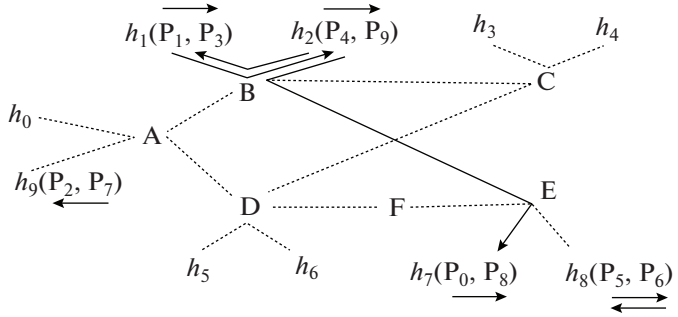

Рис. 5. Пример 2, решение второй ЦЛП-задачи при удвоенной производительности вычислительных узлов.

новить не сложно: пропускная способность соединений, связывающих коммутатор C с другими коммутаторами, недостаточна для трех входных потоков процессов P_3, P_4. Задача будет иметь решение, если добавить в систему (рис. 3б) одно из соединений (C, E) или (C, F).

Решение второй ЦЛП-задачи для тех же входных данных, в предположении, что все коммутаторы относятся к первому типу, изображено на рис. 4 (пунктиром показаны неиспользованные соединения).

Видно, что взаимодействующие процессы по возможности размещаются на смежных узлах, подключенных к одному коммутатору.

Если увеличить производительность вычислительных узлов до 20 единиц, то процессы будут сгруппированы на вычислительных узлах парами (рис. 5). Тем самым минимизируется максимальная и суммарная длина маршрутов, так как для процессов на одном узле длина маршрута нулевая.

При дальнейшем увеличении производительности узлов (30, 40, 50 единиц) происходит группирование до 3, 4, 5 процессов на узле, и происходит дальнейшее уменьшение суммарной длины маршрутов. При увеличении производительности вычислительных узлов до 60 единиц на одном из узлов размещается наибольшая слабая компонента связности графа (рис. 3а): P_0, P_1, P_3, P_4, P_8, P_9, а остальные процессы размещаются на другом узле; при этом суммарная длина маршрутов оказывается нулевой.

6. ЗАКЛЮЧЕНИЕ

Предложенный в работе подход может значительно упростить процедуру отображения сложных параллельных вычислений на распределенные системы, использующие коммуникационную среду RapidIO. Экспериментальный подбор требуемого отображения весьма трудоемок и не всегда обеспечивает оптимальный результат. Применение методов ЦЛП или других подобных методов гарантирует нахождение отображения, если оно существует, и позволяет оптимизировать его по различным параметрам.

Тем не менее, в этой области остается еще множество нерешенных вопросов, требующих дальнейших исследований. К их числу относятся вопросы масштабируемости предложенных формулировок ЦЛП-задач, дополнительные параметры оптимизации искомого отображения, более реалистичные модели параллельного вычисления и распределенной системы, наконец, задача построения распределенной системы по заданному параллельному вычислению. Остановимся на некоторых из них, представляющихся наиболее значимыми с практической точки зрения.

В целевой функции (11) предусмотрена минимизация максимальной длины маршрута, суммарной длины всех маршрутов и общее число записей в таблицах маршрутизации. К этому можно добавить:

- максимизацию минимального запаса пропускной способности по всем соединениям (чтобы избежать ситуаций, когда одни соединения используются на 100%, а другие не загружены);
- максимизацию минимального запаса производительности по всем вычислительным узлам;
- минимизацию максимальной нагрузки на коммутатор.

В формулировку задачи необходимо включить также ограничение на длину маршрутов, следующее из специфики коммуникационной среды RapidIO.

Для процессов, входящих в состав параллельных вычислений, может требоваться не только определенный уровень производительности, но и другие характеристики вычислительного узла, например, наличие определенных сопроцессоров. Кроме того, некоторые процессы должны получать входные данные от внешних устройств, поэтому они должны выполняться на вычислительных узлах, имеющих подключение к таким устройствам. Модели параллельного вычисления и распределенной системы, а также формулировку 2-й ЦЛП-задачи нетрудно расширить для учета подобных требований.

Еще один важный с практической точки зрения вопрос – ситуация, когда решение ЦЛП-задачи не существует. Причиной может быть недостаточное число соединений или их недостаточная пропускная способность, либо недостаточная производительность вычислительных узлов. Для того чтобы попытаться определить узкие места распределенной системы, можно рассмотреть решение модифицированной ЦЛП-задачи. В модифицированной задаче не используются ограничения по пропускной способности соединений и производительности узлов, но в целевой функции предусматривается минимизация максимального недостатка пропускной способности по всем соединениям и максимального недостатка производительности по всем вычислительным узлам. Рассмотрение полученного решения может дать подсказки о том, где именно следует добавить сетевые соединения, вычислительные узлы, коммутаторы.

Более радикальным решением была бы постановка задачи о построении распределенной системы под заданное параллельное вычисление, как в [23].

Пусть задано параллельное вычисление, а вместо распределенной системы определен набор

вычислительных узлов и набор коммутаторов. Для вычислительных узлов заданы уровни производительности (и, возможно, другие характеристики), а для каждого коммутатора – его тип и количество портов. Для портов заданы ограничения пропускной способности. Требуется построить:

1) множество соединений между вычислительными узлами и коммутаторами, а также между разными коммутаторами, где пропускная способность соединения определяется как минимум пропускных способностей портов, которые оно соединяет;

2) отображение процессов параллельного вычисления на вычислительные узлы полученной распределенной системы;

3) схему и таблицы маршрутизации, обеспечивающие потоки данных параллельного вычисления.

На практике распределенная система создается не из отдельных вычислительных узлов и коммутаторов, а из плат. На плате может быть расположено ограниченное количество микросхем (процессоров, микросхем памяти, коммутаторов) и разъемов для соединения с другими платами. Поэтому более реалистичной является постановка задачи, в которой заданы не отдельные вычислительные узлы и коммутаторы, а набор однотипных плат либо плат из ограниченного набора известных типов. В отличие от приведенной выше постановки задачи, требуется определить соединения не между отдельными узлами и коммутаторами, а между имеющимися платами.

Методы, подобные предложенному в работе, позволяют также решать некоторые вопросы отказоустойчивости. Например, для заданной распределенной системы и заданного параллельного вычисления (п. 4.2) можно проверить возможность альтернативного отображения параллельного вычисления при всех вариантах выхода из строя одного из элементов оборудования. Более сложная постановка задачи – построение отказоустойчивой распределенной системы под заданное параллельное вычисление, допускающей альтернативное отображение при одиночных отказах оборудования.

Таким образом, представленный в работе подход открывает возможности для целого круга исследований в области отображения параллельных вычислений на распределенные системы, а также построения распределенных систем, адаптированных к заданному параллельному вычислению.

7. БЛАГОДАРНОСТЬ

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН (проведение фундаментальных научных исследований 47 ГП) по теме № 0065-2019-0002 “Исследование и реализация программной платформы для перспективных многоядерных процессоров” (рег. № АААА-А19-119012290074-2).

СПИСОК ЛИТЕРАТУРЫ

1. RapidIO Interconnect Specification (Revision 1.3). http://www.rapidio.org/rapidio-specifications.
2. *Бобков С.Г., Задябин С.О.* Перспективные высокопроизводительные вычислительные системы промышленного применения на базе стандарта RapidIO // Электроника, микро- и наноэлектроника: сб. науч. тр. 11-й Рос. науч.-технич. конф. под ред. В.Я. Стенина. М.: Изд-во МИФИ, 2009. С. 114–121.
3. *Райко Г.О., Павловский Ю.А., Мелькановић В.С.* Технология программирования многопроцессорной обработки гидроакустических сигналов на вычислительных устройствах семейства “КОМДИВ” // Научно-технический сборник Гидроакустика. СПб.: ОАО “Концерн “Океанприбор”. 2014. Вып. 20 (2). С. 85–92.
4. *Сударева О.Ю.* Реализация алгоритма MG пакета NPB для многопроцессорного вычислительного комплекса на базе микропроцессора КОМДИВ128-РИО // Труды НИИСИ РАН. 2015. Т. 5. № 1. С. 75–87.
5. *Годунов А.Н., Солдатов В.А.* Конфигурирование многопроцессорных систем в операционной системе реального времени Багет // Программная инженерия. 2016. Т. 7. № 6. С. 243–251.
6. *Годунов А.Н., Солдатов В.А., Хоменков И.И.* Передача сообщений в коммуникационной среде RapidIO для семейства операционных систем реального времени Багет // Программная инженерия. 2020. Т. 11. № 1. С. 26–33.
7. Библиотека параллельной обработки сигналов // Труды НИИСИ РАН. 2015. Т. 5. № 1. С. 64–69.
8. *Грингауз Т.К., Онин А.Н.* Инструментальное программное обеспечение для подготовки запуска задач в мультипроцессорных комплексах реального времени // Труды научно-исследовательского института системных исследований Российской академии наук. 2015. Т. 5. № 2. С. 122–129.
9. *Павлов А.Н.* Инструментальный комплекс “РИО-оптимизатор” для разработки прикладного ПО с использованием Библиотеки параллельной обработки сигналов // Труды НИИСИ РАН. 2015. Т. 5. № 1. С. 70–74.
10. *Hoefler T., Jeannot E., Mercier G.* An overview of process mapping techniques and algorithms in high-performance computing. Wiley, 2014. *Jeannot E., Zilinskas J.* (eds.), High Performance Computing on Complex Environments. P. 75–94.
11. *Liu W., Yang L., Jiang W., Feng L., Guan N., Zhang W., Dutt N.* Thermal-Aware Task Mapping on Dynamically Reconfigurable Network-on-Chip Based Multiprocessor System-on-Chip // IEEE Transactions on Computers. 2018. V. 67. № 12. P. 1818–1834.
12. *Derin O., Kabakci D., Fiorin L.* Online task remapping strategies for fault-tolerant Network-on-Chip multiprocessors // Proceedings of the Fifth ACM/IEEE International Symposium, Pittsburgh, PA. 2011. P. 129–136.

13. *Huang K., Jiang X., Zhang X., Xiong D., Yan X.* Energy-Efficient Fault-Tolerant Mapping and Scheduling on Heterogeneous Multiprocessor Real-Time Systems // IEEE Access. 2018. V. 6. P. 57614–57630.
14. *Yang L., Liu W., Jiang W., Li M., Yi J., Sha E.H.* Application Mapping and Scheduling for Network-on-Chip-Based Multiprocessor System-on-Chip With Fine-Grain Communication Optimization // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2016. V. 24. № 10. P. 3027–3040.
15. *Tang Q., Zhu L.-H., Zhou L., Xiong J., Wei J.-B.* Scheduling directed acyclic graphs with optimal duplication strategy on homogeneous multiprocessor systems // Journal of Parallel and Distributed Computing. 2020. V. 138. P. 115–127.
16. *Tang Q., Wu S., Shi J., Wei J.* Optimization of Duplication-Based Schedules on Network-on-Chip Based Multi-Processor System-on-Chips // IEEE Transactions on Parallel and Distributed Systems. 2017. V. 28. № 3. P. 826–837.
17. *Yu Y., Prasanna V.K.* Energy-balanced task allocation for collaborative processing in networked embedded systems // ACM SIGPLAN Notices. 2003. V. 38. № 7. P. 265–274.
18. *Liu W., Yi J., Li M., Chen P., Yang L.* Energy-Efficient Application Mapping and Scheduling for Lifetime Guaranteed MPSoCs // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2019. V. 38. № 1. P. 1–14.
19. *Huang K., Zhang X., Zheng D., Yu M., Jiang X., Yan X., de Brisolara L.B., Jerraya A.A.* A Scalable and Adaptable ILP-Based Approach for Task Mapping on MP-SoC Considering Load Balance and Communication Optimization // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2019. V. 38. № 9. P. 1744–1757.
20. *Kovalov A., Lobe E., Gerndt A., Lüdtke D.* Task-Node Mapping in an Arbitrary Computer Network Using SMT Solver // International Conference on Integrated Formal Methods IFM. 2017. Integrated Formal Methods. P. 177–191.
21. *Бакулин А.А.* Проверка допустимости схемы маршрутизации в системе RapidIO // Программные продукты и системы. 2011. № 4. С. 20–23.
22. GNU Linear Programming Kit. https://www.gnu.org/software/glpk.
23. *Bobda C., Ishebabi H., Mahr P., Mbongue J.M., Saha S.K.* MeXT: A Flow for Multiprocessor Exploration // IEEE High Performance Extreme Computing Conference (HPEC). 2019. P. 1–7.