РАЗРАОТКА И ИССЛЕДОВАНИЕ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ ОПТИМИЗАЦИИ РАЗВИТИЯ ИНФРАСТРУКТУРЫ ТИПА ПОСТАВЩИК-ПОТРЕБИТЕЛЬ

А.В. Жарков, Д.Г. Пивоварт ч у к
Московский Государственный Университет им. М.В.Ломоносова
Росси я , 119991, МоскваСП-1, Ленинские гор ы, МГУ, д. 1, стр. 52, 2-й у ч ебн ыкйрп у с ВМК
E-mail: zhalex@ya.ru, dpivovartchuk@gmail.com

Под инфраструктурой “поставщик–потребитель” в данной работе понимается сеть поставок, объединяющая некоторое количество поставщиков, потребителей и дистрибьютеров. Поставщики генерируют некоторый поток, под которым, в каждом конкретном случае, можно понимать, например, энергию, энергоносители, товары и т.п. Потоки от поставщиков либо доходит непосредственно до потребителей, либо достигает распределителей, которые перераспределяют входящие в них потоки по нескольким направлениям, после чего потоки доходят до потребителей. Поток, дошедший до потребителя, поглощается им.

Целью рассматриваемой задачи является оптимизация управления развитием сети поставок от поставщика к потребителю. Сеть представляет собой набор заданных связей между поставщиками, дистрибьютерами и потребителями. Развитием сети будем понимать увелечение пропускных способностей некоторых связей. Под оптимизацией развития сети поставок понимается развитие при условии максимального удовлетворения спроса потребителей.

В данной работе даётся математическая постановка задачи при условии, что развитие сети происходит дискретно по времени, а такие параметры сети, как генерируемый поток поставщиками, принимаемый поток потребителями и пропускаемый распределителями изменяется с заранее заданными вероятностями. Приводится алгоритм решения задачи и математическое доказательство его верности. В рамках данного исследования была создана последовательная реализация алгоритма решения задачи, которая требует значительных объёмов оперативной памяти и вычислительных операций. При анализе данной реализации выяснено, что возможно создание достаточно производительного параллельного алгоритма решения задачи. Был предложен параллельный алгоритм, основанный на интерфейсе MPI. Приведены результаты тестирования производительности и масштабируемости алгоритма. Алгоритм позволяет сократить время решения задачи на время до 1400 раз меньшее при применении суперкомпьютера BlueGene/P.

RESEARCH OF PARALLEL ALGORITHM FOR OPTIMAL STRATEGY OF DEVELOPING AN INFRASTRUCTURE "SUPPLIER-CONSUMER"

A.V. Zharkov, D.G. Pivovartchuk
Moscow State University
Faculty of Computational Mathematics and Cybernetics, Leninskiye Gory,
1-52, GSP-1, Moscow, 119991, Russia
E-mail: zhalex@ya.ru, dpivovartchuk@gmail.com

An infrastructure “supplier–consumer” is represented by a directed graph. Each node of the graph represents one of the following objects: supplier, consumer, or distributer. Each couple of nodes may have a directed link (arc) that is characterized by the capacity that is the maximal possible amount of flow along the link. If a link between two nodes has the capacity that is greater than zero then there can be a flow through the link and the amount of the flow does not exceed the capacity. Optimum control of developing an infrastructure is increase link’s capacities to maximum satisfaction customer’s demand.

Calculating optimal strategy for developing an infrastructure “supplier–consumer” requires large of mathematical operations and memory. If model is used for an infrastructure of size Q, the result’s matrix has 2^Q elements. For $Q = 128$, the model has 2^{128} linear systems and the result’s matrix will require over 2^{88} TBytes of memory. We are developing a highly scalable parallel solver that uses MPI. Parallel realization is based on interface MPI "client-server". For the decision of LP subtasks we used the library of widely known algorithms of research of operations, as GNU GLPK. Program realization allows to solve problems with parameters $1 < N_1 < 32 \cdot 2^{20}$, $1 < N_2 < 2^{26}$.

We present scalability results on the IBM BlueGene/P that suggests the code can reduce reduce computation time in 1400 times.

1.1. Постановка задачи.

Под инфраструктурой "поставщик–потребитель" в данной работе понимается сеть поставок, объединяющая некоторое количество поставщиков, потребителей и дистрибьютеров. Поставщики генерируют некоторый поток, под которым, в каждом конкретном случае, можно понимать, например, энергию, энергоносители, товары и т.п. Потоки от поставщиков либо доходит непосредственно до потребителей, либо достигает распределителей, которые перераспределяют входящие в них потоки по нескольким направлениям, после чего потоки доходят до потребителей. Поток, дошедший до потребителя, поглощается им.

Целью рассматриваемой задачи является оптимизация управления развитием сети поставок от поставщика к потребителю. Сеть представляет собой набор заданных связей между поставщиками, дистрибьютерами и потребителями. Под оптимизацией развития сети поставок понимается увеличение пропускных способностей ее связей, с целью максимального удовлетворения спроса потребителей.

Задача рассматривается в предположении, что объемы поставок и спросы потребителей изменяются со временем. Развитие сети поставок должно учитывать этот факт и увеличивать пропускные способностей тех элементов сети, в направлении которых осуществляется и/или будут осуществляться основные поставки. При этом в рассматриваемой постановке задачи не предполагается, что задан единственный сценарий для изменение спроса и объема поставок в будущем. Предполагается, что для каждого потребителя/поставщика задан целый набор возможных сценариев (изменения спроса/предложения в будущем) и вероятность каждого возможного сценария. В условиях описанной неопределенности под решением задачи понимается адаптивная стратегия управления развитием сети поставок, то есть предполагается, что на каждом шаге рассматриваемого процесса наблюдаются текущие объемы поставок от всех поставщиков и текущие спросы всех потребителей и на основе этой информации принимается решение о направлениях дальнейшего развития сети.

В данной работе приводится алгоритм решения задачи и математическое доказательство его верности. Последовательный алгоритм решения данной задачи требует значительных объёмов памяти и вычислительных операций, поэтому был разработан параллельный алгоритм.

1.2. Модель инфраструктуры "поставщик–потребитель".

Сеть поставок представляет собой направленный граф. Каждая вершина графа является одним из следующих объектов: *поставщик*, *потребитель* или *дистрибьютер*. Будем обозначать: поставщиков — $S_1, \ldots, S_{N_s}$, потребителей — $D_1, \ldots, D_{N_d}$ и дистрибъютеров — $B_1, \ldots, B_{N_b}$.

Каждая пара вершин может иметь направленную связь, которая характеризуется пропускной способностью (т.е. максимальным возможным потоком через эту связь в единицу времени). Если пропускная способность связи между двумя вершинами больше нуля, то между этими вершинами возможен поток, объем которого в единицу времени не превосходит пропускной способности связи.

Направленную связь, соединяющую вершины i, j, и направленную от вершины i к вершине j будем обозначать $C_{i;\,j}$. Соответствующую этой связи пропускную способность будем обозначать $c_{i;\,j}$, а поток через эту связь будем обозначать $f_{i;\,j}$. Например, связь, соединяющую поставщика S_i и потребителя D_j обозначим $C_{s_i;\,d_j}$, ее пропускную способность

$c_{s_i;\ d_j}$, а поток $f_{s_i;\ d_j}$.

Будем считать, что выполняются следующие естественные предположения:

a) Если вершина соответствует поставщику, то эта вершина может иметь только исходящие связи с потребителями или дистрибьютерами;

b) Если вершина соответствует потребителю, то эта вершина может иметь только входящие связи от поставщиков или дистрибьютеров;

c) Если вершина соответствует дистрибьютеру, то эта вершина имеет по крайней мере одну входящую и одну исходящую связь. Дистрибьютер может иметь входящие связи от поставщиков или других дистрибьютеров и исходящие связи с потребителями или другими дистрибьютерами.

d) Каждый поставщик S_i, $i =, \ldots, N_s$, генерирует исходящий поток и характеризуется числом s_i, которое определяет объем генерируемого им потока в единицу времени.

e) Каждый потребитель D_i, $i = 1, \ldots, N_d$, принимает входящий поток и характеризуется числом d_i, которое определяет объем потока, который этот потребитель может принять в единицу времени.

f) Каждый дистрибьютер B_i, $i = 1, \ldots, N_b$, принимаемый входящий поток и превращает его в исходящий поток. Дистрибьютер характеризуется числом b_i, которое определяет объем потока, который этот дистрибьтер может перераспределить.

g) Потоки, произведенные всеми поставщиками, распределяются по сети в соответствии с некоторым правилом, которое будем назвать *статическим правилом распределения потоков*.

h) Суммарный поток, ушедший от поставщиков, в полном объеме доходит до потребителей. Причем, не предполагается, что от потребителей уходит весь поток, который он может сгенерировать.

Таким образом, получаем следующую структуру функционирования инфраструктуры за единицу времени. Каждый поставщик производит некоторый объем исходящего потока. Потоки, произведенный всеми поставщиками, распределяются по сети в соответствии со статическим правилом распределения потоков (это правило будет определено далее). Потоки, дошедшие до потребителей, принимаются ими в соответствие с их возможностями.

Обозначим множество входящих связей в вершину k через

$$C_{\ ;\ k} = \{C_{i;\ j} |\ j = k\},$$

а множество исходящих связей из вершины k через

$$C_{k;\quad} = \{C_{i;\ j} |\ i = k\}.$$

Используя введенные обозначения, запишем ограничения, которым должны удовлетворять потоки $f_{i;\ j}$ между связями:

R1) Поток вдоль связи не превосходит пропускную способность связи

$$0 \leq f_{i;\ j} \leq c_{i;\ j}. \tag{1}$$

R2) Ограничение на поток производимый поставщиком S_i, $i = 1, \ldots, N_s$,

$$\sum_{j \in C_{s_i,\ *}} f_{s_i;\ j} \leq s_i. \tag{2}$$

R3) Ограничение на поток принимаемый потребителем D_i, $i = 1, \ldots, N_d$,

$$\sum_{j \in C_{*,\ d_i}} f_{j;\ d_i} \leq d_i. \tag{3}$$

R4) Ограничение на поток принимаемый и перераспределяемый дистрибьютером B_i, $i = 1, \ldots, N_b$,

$$\sum_{j \in C_{*,\ b_i}} f_{j;\ b_i} \le b_i. \tag{4}$$

R5) Сохранение суммарного объема потока при его распределении дистрибьютером B_i, $i = 1, \ldots, N_b$,

$$\sum_{j \in C_{*,\ b_i}} f_{j;\ b_i} = \sum_{j \in C_{b_i,\ *}} f_{b_i;\ j}. \tag{5}$$

Статическое правило распределения потоков задается задачей минимизации

$$\sum_{i=1}^{N_d} \left[d_i - \sum_{j \in C_{*,\ d_i}} f_{j;\ d_i} \right] \to \min_{\{f_{i,\ j}\}}, \tag{6}$$

при ограничениях R1–R5. Задача (6) означает стремление максимально удовлетворить спрос потребителей. Отметим, что минимум в последней задачи достигается, т.к. ограничения R1–R5 задают ограниченное замкнутое множество, а функционал является непрерывной функцией своих аргументов.

1.3. Динамическая модель.

Рассмотрим заданный период времени функционирования инфраструктуры и предположим, что этот период разбит на N непересекающихся интервалов. Предполагаем, что потоки, а также состояния поставщиков, потребителей и дистрибьютеров постоянны на одном интервале времени. Для обозначения времени будем использовать переменную $t = 0, 1, \ldots, N$. Считаем, что каждая переменная, зависящая от времени, постоянная на интервале $[t, t+1)$, $t = 0, 1, \ldots, N-1$.

Начальные пропускные способности всех связей заданы

$$c_{i;\ j}(0) = c^0_{i;\ j}. \tag{7}$$

Предполагаем, что мы имеем возможность управлять пропускными способностями связей. Точнее, на каждом интервале времени мы имеем возможность увеличить пропускные способности связей, причем предполагаем, что связи, чьи пропускные способности могут быть увеличены разбиты на группы. К одной группе относятся связи, чьи пропускные способности должны быть увеличены за один и тот же интервал времени. К одной группе могут относится произвольные связи, например, одна связь или последовательность связей представляющая собой путь от некоторого поставщика к некоторому потребителю или набор несвязанных между собой связей. Предполагаем, что для каждой связи каждой группы задано то значение, на которое предполагается увеличивать пропускную способность этой связи. Таким образом, одна группа представляет собой некоторое одно мероприятие по увеличению пропускных способностей связей за один период времени.

Заданные группы связей будем обозначать $G^1, \ldots, G^M$, где каждая группа характеризуется набором

$$G^q = \{\bar{u}^q_{i;j}\}, \quad q = 1, \ldots, M, \tag{8}$$

где $\bar{u}^q_{i;j}$ – заданное значение, на которое будет увеличена пропускная способность связи $C_{i;j}$ из группы G^q. Предполагаем, что $M \ge N$, т.е. количество возможных мероприятий не меньше, чем количество интервалов времени.

Обозначим через $G(t)$ множество индексов тех групп, которые еще не были задействованы к моменту времени t, и обозначим через $\bar{G}(t)$ индексы тех групп, которые уже были задействованы к моменту времени t.

Рассмотрим управляющий параметр $q(t)$, который определяет индекс группы связей, чьи пропускные способности будет увеличены на интервале времени $[t, t+1)$. Тогда можно записать следующие соотношения

$$\begin{cases} \bar{G}(t+1) &= \bar{G}(t) \cup q(t), \\ \bar{G}(0) &= \emptyset, \end{cases} \tag{9}$$

где управляющий параметр $q(t)$ может принимать значения только из множества $G(t)$.

Множество индексов $\bar{G}(t)$ для момента времени t однозначно определяет пропускные способности всех связей в момент t следующим образом

$$c_{i;\ j}(\bar{G}(t)) = c^0_{i;\ j} + \sum_{q \in \bar{G}(t)} \bar{u}^q_{i;\ j}. \tag{10}$$

Отметим, что $\bar{G}(t)$ принимает значения из множества $\bar{\mathcal{G}}_t$, где через $\bar{\mathcal{G}}_k$ мы обозначили множество всех возможных выборок k чисел из множества $\{1, \ldots, M\}$ с точностью до их перестановок и $\bar{\mathcal{G}}_0 = \emptyset$.

Управление развитием инфраструктуры заключается в последовательном выборе на каждом интервале времени группы связей, чьи пропускные способности должны быть увеличены.

Будем предполагать, что задан набор *сценариев* (функций от времени), определяющий объемы потоков, которые будут производить каждый из поставщиков на каждом интервале времени, $s_i(t)$, объемы потоков, которые смогут принять каждый из потребителей на каждом интервале времени, $d_i(t)$, объемы потоков, которые сможет перераспределить каждый из дистрибьютеров на каждом интервале времени, $b_i(t)$. Таким образом, один сценарий представляет собой набор функций

$$(s_1(t), \ldots, s_{N_s}(t), d_1(t), \ldots, d_{N_d}(t), b_1(t), \ldots, b_{N_b}(t)).$$

Для того, чтобы описать целый набор сценариев и каждому из сценариев поставить в соответствие вероятность, воспользуемся понятием марковского процесса. Рассмотрим набор значений (s, d, b) – те значения, которые характеризуют объемы, которые генерируются поставщиками, $s \in \mathbb{R}^{N_s}$, могут быть приняты потребителями, $d \in \mathbb{R}^{N_d}$, и могут быть распределены дистрибьютерами, $b \in \mathbb{R}^{N_b}$. Зафиксируем множество возможных наборов значений

$$\mathcal{H} = \{H^e = (s^e, d^e, b^e)\}_{e=1;:::;E} \tag{11}$$

т.е. состояний рассматриваемого марковского процесса и зададим переходные вероятности между состояниями

$$p_{i+1}(s_{i+1}, d_{i+1}, b_{i+1} | s_i, d_i, b_i), \tag{12}$$

для всех $(s_{i+1}, d_{i+1}, b_{i+1})$, $(s_i, d_i, b_i) \in \mathcal{H}$. Так определенный марковский процесс будет описывать множество сценариев и вероятность каждого сценария.

Пусть задан некоторый сценарий, т.е. набор функции описывающий как будут изменяться поставки, спрос и возможности по распределению потоков. Выбирая на каждом интервале времени группу связей, чьи пропускные способности будут увеличены, в соответствии с

равенствами (9), (10), получаем изменяющиеся во времени пропускные способности связей. Все это приводит к изменению потоков от интервала к интервалу, при этом потоки должны удовлетворять ограничениям R1–R5. Получаем следующие ограничения, которые должны выполняться для каждого $t = 0, 1, \ldots, N$

$$\left\{\begin{array}{rcl} 0 \le f_{i;\,j}(t) & \le & c_{i;\,j}(\bar{G}(t)), \\ \sum\limits_{j \in C_{s_i,*}} f_{s_i;\,j}(t) & \le & s_i(t), \\ \sum\limits_{j \in C_{*,\,d_i}} f_{j;\,d_i}(t) & \le & d_i(t), \\ \sum\limits_{j \in C_{*,\,b_i}} f_{j;\,b_i}(t) & \le & b_i(t), \\ \sum\limits_{j \in C_{*,\,b_i}} f_{j;\,b_i}(t) & = & \sum\limits_{j \in C_{b_i,*}} f_{b_i;\,j}(t). \end{array}\right. \tag{13}$$

Обозначим через $F(\bar{G}(t), (s(t), d(t), b(t)))$ множество значений $\{f_{i;\,j}(t)\}$, удовлетворяющих системе (13). Отметим, что это множество замкнутое и ограниченное.

1.4. Критерий качества и задача оптимизации.

Рассмотрим следующую критерий качества

$$I(s_0, d_0, b_0) = E\left[\sum_{t=0}^{N} \beta_t \sum_{i=1}^{N_d} \left(d_i(t) - \sum_{j \in C_{*,\,d_i}} f_{j;\,d_i}(t)\right)\right] \to \min_{f f_{i,j}(tj)g_{t=0,\ldots,N};\; f q(tj)g_{t=0,\ldots,N-1}}, \tag{14}$$

где $\beta_k > 0$, $k = 0, \ldots, N$, — весовые коэффициенты упорядоченные следующим образом

$$\beta_0 < \beta_1 < \ldots < \beta_N. \tag{15}$$

Эти коэффициенты отражают тот факт, что конечное состояние развиваемой инфраструктуры важнее чем промежуточные состояния. Математическое ожидание берется по все возможным сценариям, порожденным марковским процессом (11), (12) с заданным начальным состоянием процесса (s_0, d_0, b_0). При этом, управляющий параметр $q(t)$ удовлетворяет ограничениям

$$\left\{\begin{array}{rcl} \bar{G}(t+1) & = & \bar{G}(t) \cup q(t), \quad q(t) \in G(t), \\ \bar{G}(0) & = & \emptyset, \end{array}\right. \tag{16}$$

а потоки удовлетворяют ограничениям

$$\left\{\begin{array}{rcl} 0 \le f_{i;\,j}(t) & \le & c_{i;\,j}(\bar{G}(t)), \\ \sum\limits_{j \in C_{s_i,*}} f_{s_i;\,j}(t) & \le & s_i(t), \\ \sum\limits_{j \in C_{*,\,d_i}} f_{j;\,d_i}(t) & \le & d_i(t), \\ \sum\limits_{j \in C_{*,\,b_i}} f_{j;\,b_i}(t) & \le & b_i(t), \\ \sum\limits_{j \in C_{*,\,b_i}} f_{j;\,b_i}(t) & = & \sum\limits_{j \in C_{b_i,*}} f_{b_i;\,j}(t). \end{array}\right. \tag{17}$$

Отметим, что минимизация осуществляется по множеству функций $\{f_{i;j}(t|\cdot)\}$, $q(t|\cdot)$, которые зависят от множества G незадействованных к моменту t групп и текущего состояния (s, d, b), т.е.

$$\{f_{i;j}(t|G, (s, d, b))\}, \quad q(t|G, (s, d, b)).$$

Таким образом, мы ищем оптимальное адаптивное управление развитием инфраструктуры, т.е. такое управление, которое по результатам наблюдений состояний поставщиков, потребителей и дистрибьютеров на каждом интервале времени определяет группу связей, чьи пропускные способности увеличить, из набора еще незадействованных групп. Отметим тот факт, что минимальное значение функционала достигается, т.к. аргументы функций $q(t|\cdot)$, $\{f_{i;\,j}(t|\cdot)\}$ принимают значения из конечных множеств, образ функций $q(t|\cdot)$ – конечное множество, а образ функций $\{f_{i;\,j}(t|\cdot)\}$ – замкнутое ограниченное множество $F(\bar{G}(t),(s(t),d(t),b(t)))$. Оптимальное значение функционала будем обозначать $I\ (s_0,d_0,b_0)$.

2.1. Решение задачи.

Подход к решению задачи основан на методе динамического программирования [1, 2] и линейном программирование [3].

Рассмотрим вспомогательную задачу линейного программирования, зависящую от заданного множества индексов $\bar{G}$ и заданного набора значений (s,d,b):

$$\sum_{i=1}^{N_d}\left[d_i-\sum_{j\,2\,C_{*,\;d_i}} f_{j;\;d_i}\right]\to\min_{f f_{i,j} g} \tag{18}$$

при ограничениях

$$\begin{cases} 0\le f_{i;\;j} & \le\ c_{i;\;j}(\bar{G}),\\ \sum\limits_{j\,2\,C_{s_i,*}} f_{s_i;\;j} & \le\ s_i,\\ \sum\limits_{j\,2\,C_{*,\;d_i}} f_{j;\;d_i} & \le\ d_i,\\ \sum\limits_{j\,2\,C_{*,\;b_i}} f_{j;\;b_i} & \le\ b_i,\\ \sum\limits_{j\,2\,C_{*,\;b_i}} f_{j;\;b_i} & =\ \sum\limits_{j\,2\,C_{b_i,*}} f_{b_i;\;j}. \end{cases} \tag{19}$$

Оптимальное значение функционала в последней задаче обозначим $W(\bar{G},(s,d,b))$, а оптимальное решение $\{f^{W}_{i;\,j}(\bar{G},(s,d,b))\}$.

Решение исходной задачи (14)–(17) основано на следующем утверждение:

Утверждение. Оптимальное значение функционала в задаче (14)–(17) удовлетворяет равенству

$$I\ (s_0,d_0,b_0)=V_0(\emptyset,(s_0,d_0,b_0)),$$

где функция $V_0(\bar{G},(s,d,b))$ вычисляется из следующих рекуррентных соотношений:

1) Для $\tau=0,\ldots,N-1$, всех $\bar{G}\ \in\bar{\mathcal{G}}$ и всех $(s\ ,d\ ,b\ )\in\mathcal{H}$

$$V\ (\bar{G}\ ,(s\ ,d\ ,b\ ))=\beta\ W(\bar{G}\ ,(s\ ,d\ ,b\ ))+$$

$$+\min_{q_\tau 2 G_\tau}\left\{\sum_{(s_{\tau+1};d_{\tau+1};b_{\tau+1})2H} p\ _{+1}(s\ _{+1},d\ _{+1},b\ _{+1}|s\ ,d\ ,b\ )V\ _{+1}(\bar{G}\ \cup q\ ,(s\ _{+1},d\ _{+1},b\ _{+1}))\right\}. \tag{20}$$

2) Для всех $\bar{G}_N\in\bar{\mathcal{G}}_N$ и всех $(s_N,d_N,b_N)\in\mathcal{H}$

$$V_N(\bar{G}_N,(s_N,d_N,b_N))=\beta_N W(\bar{G}_N,(s_N,d_N,b_N)). \tag{21}$$

Доказательство.

Приведем схему доказательства. Следуя методу динамического программирования рассмотрим семейство задач. Это семейство будем параметризовать моментом времени τ,

набором индексов $\bar{G}$ задействованных к моменту времени τ групп, состоянием поставщиков, потребителей и дистрибьютеров $H = (s, d, b)$ в момент времени τ. Одна задача семейства представляет собой следующую задачу минимизации на отрезке времени $[\tau, N]$:

$$E_{H_\tau}\left[\sum_{t=}^{N}\beta_t\sum_{i=1}^{N_d}\left(d_i(t)-\sum_{j\,2\,C_{*,\,d_i}} f_{j;\,d_i}(t)\right)\right]\to \min_{f f_{i,j}(tj)g_{t=\tau,\ldots,N};\ f q(tj)g_{t=\tau,\ldots,N-1}}, \tag{22}$$

где математическое ожидание берется по всем сценариям на отрезке времени $[\tau, N]$, начинающимся из состояния H, при следующих ограничениях на управление $q(t)$

$$\begin{cases} \bar{G}(t+1) &= \bar{G}(t)\cup q(t), \quad q(t)\in G(t), \\ \bar{G}(\tau) &= \bar{G}, \end{cases} \tag{23}$$

и на потоки

$$\begin{cases} 0\le f_{i;\,j}(t) &\le c_{i;\,j}(\bar{G}(t)), \\ \sum\limits_{j\,2\,C_{s_i,*}} f_{s_i;\,j}(t) &\le s_i(t), \\ \sum\limits_{j\,2\,C_{*,\,d_i}} f_{j;\,d_i}(t) &\le d_i(t), \\ \sum\limits_{j\,2\,C_{*,\,b_i}} f_{j;\,b_i}(t) &\le b_i(t), \\ \sum\limits_{j\,2\,C_{*,\,b_i}} f_{j;\,b_i}(t) &= \sum\limits_{j\,2\,C_{b_i,*}} f_{b_i;\,j}(t). \end{cases} \tag{24}$$

Оптимальное значение в этой задаче минимизации обозначим $V(\bar{G}, H)$.

Для заданного адаптивного управления $u = (\{f_{i;j}(t|\cdot)\}_{t=0;\ldots;N},\ \{q(t|\cdot)\}_{t=0;\ldots;N\ \ 1})$ обозначим $u = (\{f_{i;j}(t|\cdot)\}_{t=\ ;\ldots;N},\ \{q(t|\cdot)\}_{t=\ ;\ldots;N\ \ 1})$.

Рассмотрим момент времени $\tau\in\{0,1,\ldots,N-1\}$, множество индексов $\bar{G}\in\bar{\mathcal{G}}$ и состояние $H = (s, d, b)\in\mathcal{H}$. Значение функции $V(\bar{G}, H)$, согласно ее определению, имеет вид

$$V(\bar{G}, H) = \min_{u^\tau} E_{H_\tau}\left[\sum_{t=}^{N}\beta_t\sum_{i=1}^{N_d}\left(d_i(t)-\sum_{j\,2\,C_{*,\,d_i}} f_{j;\,d_i}(t)\right)\right],$$

где задача минимизации рассматривается при ограничениях (23)–(24). Адаптивное управление в этой задаче можно представить в виде

$$u = ((\{f_{i;j}\},\ q),\ u^{+1}),$$

где управляющие параметры $\{f_{i;j}\}\in F = F(\bar{G}, H)$, $q\in G$ соответствуют действиям в момент времени τ в случае множества индексов $\bar{G}$ и состояния H. Пользуясь определением математического ожидания, получаем

$$V(\bar{G}, H) = \min_{((f f^{\tau}_{i,j}g2F_\tau;\ q_\tau 2G_\tau);\ u^{\tau+1})}\Bigg\{ \beta\sum_{i=1}^{N_d}\left(d_i(\tau)-\sum_{j\,2\,C_{*,\,d_i}} f_{j;\,d_i}(\tau)\right)+\sum_{H_{\tau+1}2H} p^{\ +1}(H^{\ +1}|H)\times \times E_{H_{\tau+1}}\left[\sum_{t=\ +1}^{N}\beta_t\sum_{i=1}^{N_d}\left(d_i(t)-\sum_{j\,2\,C_{*,\,d_i}} f_{j;\,d_i}(t)\right)\right]\Bigg\}.$$

Так как первое слагаемое не зависит от q и u^{+1}, можно записать

$$V(\bar{G}, H) = \beta \min_{ff_{i,j}^{\tau} g 2 F_\tau} \left\{ \sum_{i=1}^{N_d} \left(d_i(\tau) - \sum_{j 2 C_{*,\, d_i}} f_{j;\, d_i}(\tau) \right) \right\} +$$

$$+ \min_{q_\tau 2 G_\tau} \left\{ \sum_{H_{\tau+1} 2 H} p_{+1}(H_{+1}|H) \min_{u^{\tau+1}} E_{H_{\tau+1}} \left[\sum_{t=+1}^{N} \beta_t \sum_{i=1}^{N_d} \left(d_i(t) - \sum_{j 2 C_{*,\, d_i}} f_{j;\, d_i}(t) \right) \right] \right\}.$$

Задача минимизации первого слагаемого совпадает со вспомогательной задачей линейного программирования (18)–(19), оптимальное значение которой мы обозначили $W(\bar{G}, H)$, а задача минимизации последнего множителя совпадает с задачей введенного семейства задач, принимая во внимание, что $\bar{G} \cup q \in \bar{\mathcal{G}}_{+1}$. Получаем

$$V(\bar{G}, H) = \beta\, W(\bar{G}, H) + \min_{q_\tau 2 G_\tau} \left\{ \sum_{H_{\tau+1} 2 H} p_{+1}(H_{+1}|H) V_{+1}(\bar{G} \cup q, H)) \right\}.$$

Таким образом, равенство (20) доказано. Равенство (21) следует непосредственно из определения семейства задач (22)–(24). Из определения семейства задач (22)–(24) следует, что исходная задача совпадает с задачей семейства в случае параметров $\tau = 0$, $\bar{G}_0 = \emptyset$, (s_0, d_0, b_0), поэтому

$$I(s_0, d_0, b_0) = V_0(\emptyset, (s_0, d_0, b_0)).$$

Что и требовалось доказать.

Следствие. Оптимальное адаптивное управления в задаче (14)–(17) имеет следующий вид: пусть в момент времени τ индексы уже включенных групп связей задаются множеством $G \in \bar{\mathcal{G}}$ и наблюдается состояние $(s, d, b) \in \mathcal{H}$, тогда

$$f_{i;\, j}(\tau|G, (s, d, b))) = f_{i;\, j}^{W}(\bar{G}, (s, d, b)), \tag{25}$$

$$q(\tau|G, (s, d, b))) =$$

$$= arg \min_{q_\tau 2 G_\tau} \left\{ \sum_{(s_{\tau+1}; d_{\tau+1}; b_{\tau+1}) 2 H} p_{+1}(s_{+1}, d_{+1}, b_{+1}|s, d, b) V(\bar{G} \cup q, (s_{+1}, d_{+1}, b_{+1})) \right\}. \tag{26}$$

2.2. Алгоритм построения оптимального управления.

Опишем схему нахождения оптимального решения задачи. Алгоритм включает в себя следующие шаги:

1) Для каждого $\tau = 0, 1, \ldots, N$, множества $\bar{G} \in \bar{\mathcal{G}}$ и набора $(s, d, b) \in \mathcal{H}$ решаем вспомогательную задачу линейного программирования (18)–(19) и находим значения $W(\bar{G}, (s, d, b))$.

2) Для каждого $\tau = 0, 1, \ldots, N$, множества $\bar{G} \in \bar{\mathcal{G}}$ и набора $(s, d, b) \in \mathcal{H}$ на основе рекуррентных соотношений (20)–(21) вычисляем значения $V(\bar{G}, (s, d, b))$.

3) Пусть наблюдается последовательность состояний $\{(s_t, d_t, b_t)\}_{t=0;\ldots;N}$. Оптимальное управление строится следующим образом. Для начального момента времени задаем $\bar{G}(0) = \emptyset$

и $G(0) = \{1, \ldots, M\}$,

$$q_0 = arg \min_{q_0 2 f 1;:::;M g} \left\{ \sum_{(s_1;d_1;b_1) 2 H} p_1(s_1, d_1, b_1)|s_0, d_0, b_0) V_0(q_0, (s_1, d_1, b_1)) \right\}.$$

Для момента времени $t = 1$, получаем $\bar{G}(1) = \{q_0\}$ и $G(1) = \{1, \ldots, M\} \setminus \{q_0\}$, значение q_1 вычисляем по формуле (26), и т.д. Для момент времени $t = k$, получаем $\bar{G}(k) = \{q_0, \ldots, q_{k\ \ 1}\}$ и $G(k) = \{1, \ldots, M\} \setminus \{q_0, \ldots, q_{k\ \ 1}\}$,

$$q_k = arg \min_{q_k 2 G(k)} \left\{ \sum_{(s_{k+1};d_{k+1};b_{k+1}) 2 H} p_{k+1}(s_{k+1}, d_{k+1}, b_{k+1})|s_k, d_k, b_k) V_k(q_k, (s_{k+1}, d_{k+1}, b_{k+1})) \right\}.$$

Таким образом, получаем оптимальную последовательность $(q_0, \ldots, q_{N\ \ 1})$ включения групп связей, соответствующую заданной наблюдаемой последовательности $\{(s_t, d_t, b_t)\}_{t=0;:::;N}$.

3.1. Последовательная реализация алгоритма.

Была выполнена последовательная реализация описанного алгоритма и проведен численный эксперимент на вычислительной машине Intel Pentium Dual 1.86 GHz с 1 GB оперативной памяти. Время решения задачи зависит от двух основных параметров входных данных: $N_1 = N * (N + N_s + N_d + 2 * N_b)$ – число неравенств в подзадаче линейного программирования, которую необходимо решить, чтобы получить значение одного элемента матрицы W , $N_2 = E * 2^Q$ – число элементов матрицы W. В реализации

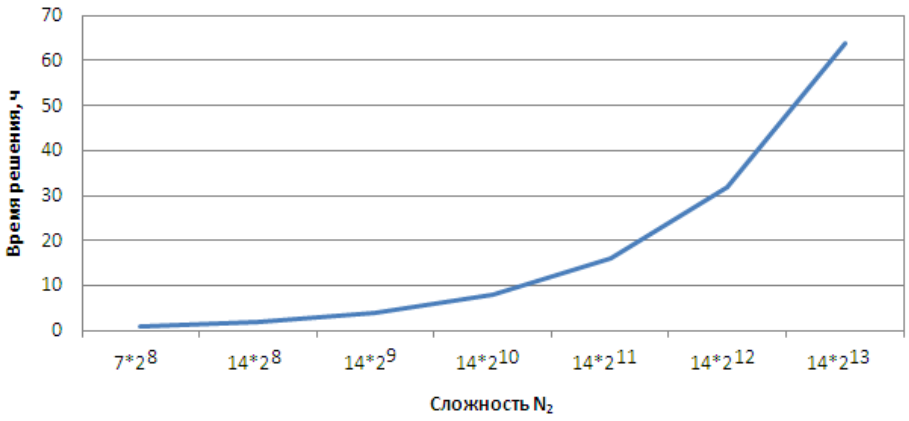

Рис. 1: Время решения задачи на однопроцессорном компьютере

алгоритма использовалась библиотека GNU GLPK 4.7 для решения подзадач линейного программирования симплекс-методом, сборка производилась компилятором GCC 3.4.2. Результаты численного эксперимента для входных данных с параметрами $N = 5500$, $N_s = 60$,$N_d = 50$, $N_b = 50$, $K = 10$, $E = 7..14$, $Q = 8..13$ приведены на графике (рис. 1).

Время решения задачи с параметрами $N_1 = 31.405.000$, $N_2 = 14 * 2^{13}$ ($E = 14$, $Q = 13$)составляет порядка 64 часов, и прогнозируемое время решения такой же задачи при значении количества моментов времени $Q = 30$ (например, прогноз на месяц по дням) составляет 957 лет. В связи с большой вычислительной сложностью алгоритма была разработана параллельная реализация.

3.2. Параллельная реализация алгоритма.

Профилирование кода последовательной реализации на задачах с достаточно большими показателями N1>10.000 и N2 (E>1, Q>5) показало, что 98% времени решения задачи занимает вычисление матрицы W, т.е. решение подзадач линейного программирования, 2% времени производится ввод и подготовка данных. Для того, чтобы параллельная реализация оказалась быстрее последовательной, необходимо удачно решить проблему загрузки, синхронизации и балансировки параллельных потоков. Из двух основных групп методов решения проблем балансировки: статических и динамических, был выбран метод динамической балансировки, ввиду недостаточной априорной информации о процессе для статического планирования. Метод динамического планирования удобен при проведении вычислений, распадающихся на большое количество однотипных задач, каждая из которых решается независимо от остальных. Передачи данных между такими задачами нет, а значит, полностью отсутствует необходимость их взаимной синхронизации. Выделяется один управляющий процессор, все остальные используются в качестве счетных узлов. Каждый счетный процессор выполняет решение системы линейных неравенств с соответствующими локальными параметрами. На управляющий процессор возложены функции распределения элементарных заданий по обрабатывающим процессорам и сбора полученных результатов. В начале очередного шага каждый из счётных процессоров ожидает новую задачу, обрабатывает ее, возвращает результат и снова переходит к ожиданию очередного задания, пока не получит в ответ сообщение, что все системы неравнств решены. Отсутствие необходимости выполнения синхронизации между элементарными задачами позволяет передавать разным процессорам разное количество задачпо мере окончания обработки данных. Тем самым решается проблема равномерной загрузки процессоров, даже если время решения системы неравенств для разных узлов или производительность процессоров сильно отличаеюся. При вычислении элементов матрицы W необходимо решить N_2 независимых подзадач линейного программирования. Входные данные каждой подзадачи не зависят от решения других подзадач и являются входными данными основной задачи, за исключением двух параметров G(набор групп) и номер e набора SDB. Решение о загрузке процессоров принимается в одном отдельном управляющем процессе следующим образом: при освобождении очередного процессора на него назначается следующая по номеру задача из упорядоченного списка по G и e (G, e). Реализация выполнена на основе программного интерфейса MPI. После загрузки управляющим процессом данных из входного файла они рассылаются на каждый процессор командой коллективного взаимодействия MPI_BCAST. Затем каждый управляющий процесс входит в цикл ожидания запроса от вычислительных процессов и при поступлении запроса отсылает номер очередной задачи из списка (G, e). При поступлении ответа решения подзадачи линейного программирования он сохраняется управляющим процессом в матрице W. Таким образом, передача данных между процессорами минимизирована, и для решения одной подзадачи требуется пересылка не более $4 * sizeof(float)$ байт, что происходит существенно быстрее, чем само решение подзадачи симплекс методом. Описанное решение проблем загрузки, синхронизации и балансировки параллельных потоков позволило добиться достаточно высокой эффективности параллельного алгоритма. Сборка производилась компилятором mpicxx на системе IBM BlueGene/P. Протестированы модельные задачи с параметрами N_1 от $4{\cdot}10^5$ ($N = 40$,$N_s = 10$,$N_d = 10$,$N_b = 10$) до $34{\cdot}10^6$ ($N = 4045$, $N_s = 1000$, $N_d = 1000$,$N_b = 1000$) и N_2 от 2 ($E = 1$, $Q = 1$) до 67.108.864 ($E = 1$, $Q = 26$). Запуск проводился на числе процессоров от 2 до 2048.

По результатам запуска алгоритма на тестовых моделях были получены следующие усредненные графики эффективности и масштабируемости параллельных вычислений,

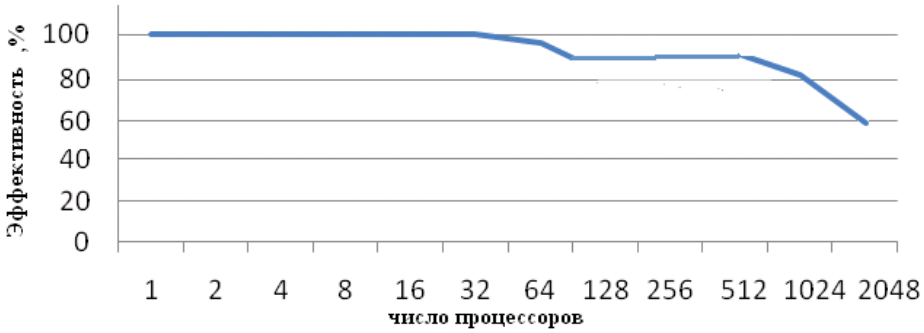

Рис. 2: Эффективность использования процессоров

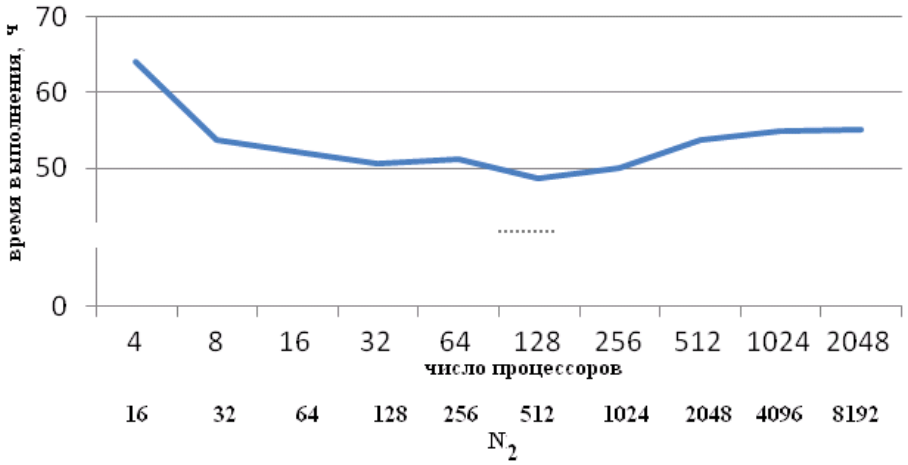

Рис. 3: Масштабируемость.Зависимость времени выполнения от размера задачи и числа процессоров

показанные на рис. 2 и рис. 3, в соответствии с [4, 5].

4. Заключение

1. Программная реализация позволяет решать задачи с параметрами $1 < N_1 < 32 \cdot 2^{20}$ и $1 < N_2 < 2^{26}$;

2. Максимальное достигнутое ускорение на Bluegene/P на задаче $n_1 = 13107200$, $n_2 = 8192$ составило 1444 при запуске на 2048 процессорах, эффективность использования процессоров на этой задаче 70%;

3. Прогнозируемое время решения задачи с параметрами $n_1 = 30 \cdot 20^20$, $n_2 = 2^{26}$ на 1 процессоре составляет 36 лет, применение 2048 процессоров позволит сократить время решения до 9 дней;

4. Время решения задачи незначительно изменяется при увеличении её вычислительной сложности в k раз с одновременным увеличением числа процессоров в k раз (рис. 3), т.е. задача хорошо масштабируется.

Список литературы

[1] Беллман Р. Динамическое программирование. М.: Издательство иностранной

литературы, 1960.

[2] Габасов Р., Кириллова Ф.М. Основы динамического программирования. Минск: Издательство Белорусского университета, 1975.

[3] Васильев Ф.П., Иваницкий А.Ю. Линейное программирование. М.: Факториал, 2003.

[4] Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002.

[5] Антонов А.С. Введение в параллельное программирование (методическое пособие). М.: НИВЦ МГУ, 2002.