ISSN 2306-1561
Automation and Control in Technical Systems (ACTS)
2015, No 2, pp. 123-145.
DOI: 10.12731/2306-1561-2015-2-11

Models and Methods of Organization of Parallel Computing Systems and Distributed Data Processing on the Basis of the Neuroprocessor

Vitaly Aleksandrovich Romanchuk

Russian Federation, Ph.D., Senior Lecturer, Department of «Computer Science and Engineering».

Ryazan State University named for S.A. Esenin, 390000, Russian Federation, Ryazan, Svobody Str., 46, aud. 41. Tel.: +7 (4912) 28-05-00, http://www.rsu.edu.ru

v.a.romanchuk@yandex.ru

Abstract. This article offers solutions for the organization of parallel computing systems and distributed data processing on the basis of the neuroprocessor using set-theoretic approach, comprising the following stages: development of the methodological foundations of the breaking of neurosecretory on many routines, development of methodological foundations of the study of links between routines, classification and selection neuroprocessor structures. Shown developed software tools for the organization of complex systems on the basis of the neuroprocessor. The reported study was funded by RFBR according to the research project №14-07-00261 a.

Keywords: neuroprocessor, model of description, function, parallel systems, computer system.

ISSN 2306-1561
Автоматизация и управление в технических системах (АУТС)
2015. – № 2. – С. 123-145.
DOI: 10.12731/2306-1561-2015-2-11

УДК 004.383.3

Модель и методы организации вычислительных систем параллельной и распределенной обработки данных на базе нейропроцессоров

Романчук Виталий Александрович

Российская Федерация, кандидат технических наук, старший преподаватель кафедры «Информатики и вычислительной техники».

ФГБОУ ВПО «Рязанский государственный университет имени С.А. Есенина», 390000, Российская Федерация, г. Рязань, ул. Свободы, 46, ауд. 41, Тел.: +7 (4912) 28-05-00, http://www.rsu.edu.ru

v.a.romanchuk@yandex.ru

Аннотация. Предлагается решение задачи организации вычислительных систем параллельной и распределенной обработки данных на базе нейропроцессоров с использованием теоретико-множественного подхода, включающее следующие этапы: разработка методологических основ разбиения нейромикропрограммы на множество подпрограмм, разработка методологических основ исследования связей между подпрограммами, классификация и выбор нейропроцессорных структур. Показаны разработанные программные средства организации сложных вычислительных систем на базе нейропроцессоров. Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта №14-07-00261 а.

Ключевые слова: нейропроцессор, модель описания, параллельная система, вычислительная система, распределенная система, облачная система.

1. Введение

В настоящее время для процессоров наступил так называемый ”технологический предел”, означающий, что они достигли максимального уровня повышения быстродействия. Промышленные разработки, в основном, направлены на повышение числа процессоров на кристалле, в данной статье предлагается в качестве выхода из данной ситуации новая элементная база – нейрокомпьютеры [3].

Нейрокомпьютер – устройство переработки информации на основе принципов работы естественных нейронных систем.

Принципиальными преимуществами нейрокомпьютера являются:

1. Высокий уровень параллелизма операций. Проблема эффективного параллелизма изучается долгое время для компьютеров, работающих в соответствии с принципами фон Неймана, но задача до сих пор не решена в полной мере, что отражается в законе Гроша и гипотезе Минского. Закон Гроша определяет, что производительность одного процессора увеличивается пропорционально квадрату его стоимости (на практике закон выполняется до определенной экспериментами границы). Гипотеза Минского утверждает, что в параллельной системе с n процессорами производительность каждого из которых равно 1, общая производительность растет лишь как $\log_2 n$. Для различных классов задач строятся максимально параллельные алгоритмы решения, использующие какую-либо абстрактную архитектуру (парадигму) мелкозернистого параллелизма, а для конкретных параллельных компьютеров создаются средства реализации параллельных процессов заданной абстрактной архитектуры. Нейроинформатика предоставляет универсальные мелкозернистые параллельные архитектуры для решения различных классов задач, для которых строится абстрактная нейросетевая реализация алгоритма решения, которая затем реализуется на конкретных параллельных вычислительных устройствах - нейрокомпьютерах [4].

2. Надежность системы, которая определяется свойствами:

- однородность системы – элементы нейросети одинаковы и просты;
- избыточное количество связей, при котором можно пренебречь «ненадежными элементами»,
- «голографичность» – свойство сохранения свойств и работоспособности системы при разрушении части системы.

3. Высокая энергоэффективность, которая достигается за счет использования однотипных аппаратных ресурсов, предназначенных для эмуляции элементарных нейронов.

Одна из проблем нейрокомпьютеров – невысокая производительность нейропроцессорных устройств в связи с небольшой частотой нейрочипов (30-320 МГц). Одним из способов ее решения является организация многопроцессорных систем. В настоящее время в области нейропроцессорных технологий ведутся исследования в части многопроцессорности [3, 7], разработаны модули, включающие несколько процессоров с различными связями [плата ВМ1, плата МЦ4.04, плата МЦ4.13 (мезонин), МЦ9.01, разработанные в НТЦ «Модуль»; вычислительные модули SMT302, SMT344, SMT313, SMT315, SMT316 на базе 1,2 и 4 модулей семейства TMS320C4x, разработанные фирмой Sundance].

Проблемой, мешающей созданию эффективных мультимикропроцессорных структур на базе нейропроцессоров стало то, что проектирование и организация и специализированных параллельных и распределенных многопроцессорных систем на базе нейропроцессоров являются очень трудоемким и сложным процессом, так как, в отличие от обычных процессоров, для эффективной работы нейропроцессорных систем нет разработанной теории, методов и алгоритмов и программных средств [7].

Цель работы: исследование функционирования многопроцессорных систем на базе нейропроцессоров и разработка модели описания и методов организации вычислительных систем параллельной и распределенной обработки данных на базе нейропроцессоров.

2. Использование теоретико-множественного подхода для решения задачи организации вычислительных систем параллельной и распределенной обработки данных на базе нейропроцессоров

Пусть $A^{(j)}$ – некоторый j-й алгоритм обработки информации, предназначенный для реализации на многопроцессорной системе на базе нейропроцессоров.

Согласно описанию функционирования нейрокомпьютера, алгоритм $A^{(j)}$ представляет собой кортеж операций вида $y = f(g) = f(\sum_{i=1}^{n} a_i x_i + a_0)$ и вспомогательных операций. Введем множество операций $O = \{O_1, O_2, ..., O_i, ..., O_o\}$. С учетом введенного множества операций этот алгоритм обработки информации представляет собой кортеж, состоящий из операций $O_1, O_2, ..., O_m, ..., O_M$ длиной $L_j = |A^{(j)}|; j = \overline{1, N}$, т.е.:

$$A^{(j)} = < O_1, O_2, ..., O_m, ..., O_M >; j = \overline{1, N}. \quad (1)$$

Исходя из математического смысла нейрокомпьютерной обработки, множество операций нейрокомпьютера рационально разделить на 2 подмножества:

1. Множество операций первого типа, представляющих математическую модель формального нейрона, и состоящее из одной операции $O^{(1)} = \{O_1^{(1)}\}$:

$$O_1^{(1)} = f(\sum_{m=1}^{n} a_m x_m + a_0) \quad (2)$$

2. Множество вспомогательных операций второго типа: обмена данными, управления подготовки данных и др.

$$O^{(2)} = \{O_1^{(2)}, O_2^{(2)}, ..., O_i^{(2)}, ..., O_o^{(2)}\} \quad (3)$$

Таким образом, множество операций для нейрокомпьютера может быть описано следующим образом:

$$O = O_1^{(1)} \cup O^{(2)} \quad (4)$$

Тогда алгоритм (1) можно записать следующим образом:

$$A^{(j)} = < O_1, O_2, ..., O_m, ..., O_M >; O_m = \{O_1^{(1)}, O_1^{(2)}, O_2^{(2)}, ..., O_o^{(2)}\}; j = \overline{1, N} \quad (5)$$

Пусть число команд $O_1^{(1)}$ в кортеже равно O_{NP}. Тогда число команд $O^{(2)}$ в кортеже равно $L_j - O_{NP}$.

На этапе разработки программы каждой операции ставится в соответствие микрокоманда процессора $MK_p; \forall p = \overline{1, K_i}$, где $MK = \{MK_1, MK_2, ..., MK_i, ..., MK_I\}$ –

множество микрокоманд, написанных на внутреннем языке нейропроцессора; K_i – минимальное количество микрокоманд, необходимое для реализации операции O_m.

Множество микрокоманд также можно разделить на 2 подмножества:

$$MK = MK^{(1)} \cup MK^{(2)}, \quad (6)$$

В этом случае для каждой операции первого типа:

$$\forall O_m \in O^{(1)} : O_1^{(1)} \to \{MK_p^{(1)}\}; \forall p = \overline{1, K_i}; \forall m = \overline{1, O_{NP}}, \quad (7)$$

Нейрокомпьютер – высокопараллельное устройство, поэтому одна микрокоманда может соответствовать кортежу операций первого типа:

$$\forall MK_p^{(1)} \in MK^{(1)} : MK_p^{(1)} \to < O_1^{(1)}, O_2^{(1)}, ..., O_i^{(1)}, ..., O_R^{(1)} >; \forall p = \overline{1, K_i} \quad (8)$$

Для операций второго типа:

$$\forall O_l \in O^{(2)} : O_l^{(2)} \to \{MK_p^{(2)}\}; \forall p = \overline{1, K_i}; \forall l = \overline{1, L_j - O_{NP}}, \quad (9)$$

В зависимости от решения задач (7) и (9) на следующем этапе необходимо каждому j-му алгоритму обработки поставить в соответствие некоторую программу $PR^{(j)}$, т.е. требуется определить некоторое отображение φ:

$$\varphi : A^{(j)} \to PR^{(j)}, \; j = \overline{1, N}. \quad (10)$$

Под программой $PR^{(j)}$ понимается кортеж микрокоманд:

$$PR^{(j)} = < MK_1, MK_2, ..., MK_i, ..., MK_I >; MK_i = \{MK_i^{(1)}, MK_i^{(2)}\}; i = \overline{1, I};, \quad (11)$$

длиной $L^{(j)} = \left| PR^{(j)} \right|$.

Следующим этапом является разделение кода программы $\boldsymbol{PR^{(j)}}$ на множество подпрограмм $\{RO_l^{(j)}\}; l = \overline{1, L}$ и загрузка подпрограмм на каждый НПВМ.

Математически данный этап можно описать следующим образом:

$$PR^{(j)} \to \{RO_l^{(j)}\}; l = \overline{1, L}; \forall j = \overline{1, N}. \quad (12)$$

Далее необходимо исследование связей, то есть определение зависимостей между нейропроцессорными вычислительными модулями (НПВМ) нейропроцессорной вычислительной системы (НПВС), на которые загружены подпрограммы из множества $\{RO_l^{(j)}\}; l = \overline{1, L}; \; j = \overline{1, N}$.

На следующем этапе необходимо определить вид структуры НПВС. Введем понятие структуры S_w, под которым понимается отношение параллельности выполнения подпрограмм RO_{i1}, RO_{i2} между i_1-м и i_2-м процессорными модулями $R_{i1} S_w R_{i2}; \forall R_{i1}, R_{i2} \in PR^{(j)}$. Отношение параллельности понимается как выполнение одновременно двух или несколько подпрограмм на разных процессорных модулях.

Следовательно, необходимо определить структуру $S_w \in S; w = \overline{1, W}$ из множества структур $S = \{S_1, S_2, ..., S_w ..., S_W\}$, позволяющую некоторой j-й программе $PR'^{(j)}$ (12) поставить в соответствие множество подпрограмм $\{RO_l^{(j)}\}; l = \overline{1, L}$:

$$S_w : PR^{(j)S_w} \rightarrow \{RO_l^{(j)}\}; l = \overline{1, L}; \forall j = \overline{1, N}\ . \qquad (13)$$

Эта структура описывается типом, числом процессорных модулей q и связями между ними, что, в свою очередь, позволит получить оценки производительности для НПВС конкретной структуры.

3. Разработка методологических основ рационального разбиения нейромикропрограммы на множество подпрограмм

В данном пункте на основании алгебраического подхода предлагается методика разбиения программы $PR'^{(j)}$ на множество подпрограмм $\{RO_l^{(j)}\}; l = \overline{1, L};\ j = \overline{1, N}$ исходя из входных данных в виде числа НПВМ q [12].

Ранее было определено, что под нейромикропрограммой $PR^{(j)}$ понимается кортеж микрокоманд (12), где микрокоманда может быть двух типов (11):

1. Микрокоманда $MK^{(1)}$ реализации операций первого типа $O_1^{(1)} = f(\sum_{m=1}^{n} a_m x_m + a_0)$, представляющая программный код эмуляции нейрона и состоящая из одной операции.

2. Микрокоманда $MK^{(2)}$ реализации вспомогательных операций второго типа для обмена данными, управления подготовки данных и др. Команда предполагает широкий набор возможных операций: операции сложения, деления, адресации, условных переходов, циклов и др.

Исходя из принципов функционирования и назначения нейрокомпьютера, количество и вид команд $MK^{(2)}$ зависит от кортежа команд $MK^{(1)}$, то есть каждая произвольно взятая команда $MK_n^{(2)}$ зависит от некоторой команды $MK_j^{(1)}$, то есть:

$$\forall MK_n^{(2)} \in RP^{(j)} : MK_n^{(2)} = f(MK_j^{(1)}); i = \overline{1, N_2}\ , \qquad (14)$$

где N_2 – количество операций второго типа.

Каждой команде $MK^{(1)}$ может соответствовать одна или несколько команд $MK^{(2)}$, подготавливающих данные.

Исходя из принципов функционирования нейрокомпьютера и отношения (14), в дальнейших рассуждениях считаем команды $MK^{(2)}$ не влияющими на функционирование нейропроцессорной системы и не рассматриваем отдельно.

Рассмотрим отношение двух произвольно взятых микрокоманд $MK_k^{(1)}$ и $MK_l^{(1)}$, каждая из которых представляет собой комплекс не связанных между собой команд $MK^{(1)} = \{MK_1^{(1)}, MK_2^{(1)}, ..., MK_i^{(1)}, ..., MK_{mk1}^{(1)}\}$, число которых зависит от конкретного нейрокомпьютера. Рассматривая две команды в виде «черного ящика», имеем:

- одинаковое функциональное назначение – эмуляция нейрона;
- одинаковое время выполнения команды, равное одному такту нейропроцессора;
- воздействие на один и тот же набор аппаратных ресурсов;

- аппаратная реализация нейрокомпьютера предполагает один и тот же путь потока данных и потока команд.

Введем понятие функционального равенства микрокоманд $MK_k^{(1)} = MK_l^{(1)}$, под которым будет подразумеваться равенство времени исполнения команды $T_{MK_k^{(1)}} = T_{MK_l^{(1)}}$ и равенство используемых командой аппаратных ресурсов $R_{MK_k^{(1)}} = R_{MK_l^{(1)}}$, т.е:

$$MK_k^{(1)} = MK_l^{(1)} = \begin{cases} T_{MK_k^{(1)}} = T_{MK_l^{(1)}} \\ R_{MK_k^{(1)}} = R_{MK_l^{(1)}} \end{cases}. \quad (15)$$

В этом случае справедливо утверждение, что отношение E двух произвольно взятых микрокоманд $MK_k^{(1)}$ и $MK_l^{(1)}$, удовлетворяющее (15) есть отношение эквивалентности. Отношение эквивалентности – это бинарное отношение, для которого выполнены следующие условия: рефлексивность, симметричность, транзитивность.

Рассмотрим каждое из этих условий:

Любая микронейрокоманда $MK_k^{(1)}$, являющаяся отражением модели нейрона, может выполняться параллельно самой себе, что подразумевает сама архитектура нейросети и нейрокомпьютера, т.е. $MK_k^{(1)} EMK_k^{(1)}$. Таким образом, справедливо условие рефлективности команд $\forall MK_k^{(1)} \in PR^{(j)}; k = \overline{1.I}$.

Если команда $MK_k^{(1)}$ функционально равна команде $MK_l^{(1)}$, то и команда $MK_k^{(1)}$ функционально равна команде $MK_l^{(1)}$ и одновременно с ней может выполняться, что предполагается исходя из парадигмы нейрокомпьютинга т.е.:

$$\forall MK_k^{(1)}, MK_l^{(1)} \in PR^{(j)} : MK_k^{(1)} ERO_k \rightarrow MK_l^{(1)} EMK_k^{(1)}. \quad (16)$$

Другими словами выполняется условие симметричности двух нейромикрокоманд.

Если команда $MK_k^{(1)}$ функционально равна команде $MK_l^{(1)}$, а команда $MK_l^{(1)}$ функционально равна $MK_m^{(1)}$, тогда команда $MK_k^{(1)}$ функционально равна $MK_m^{(1)}$ и параллельна команде $MK_m^{(1)}$, т.е.

$$\forall MK_k^{(1)}, MK_l^{(1)} \in PR^{(j)} \in E : MK_k^{(1)} EMK_l^{(1)}, MK_l^{(1)} EMK_m^{(1)} \rightarrow MK_k^{(1)} EMK_m^{(1)}. \quad (17)$$

Таким образом, справедливо условие транзитивности нейромикрокоманд.

Введем понятие сегмента SG нейромикропрограммного кода, подразумевающего кортеж команд $MK^{(1)}$, и связанных вспомогательных команд подготовки данных $MK^{(2)}$:

$$SG_i^{(j)} = << MK_1^{(2)}, MK_2^{(2)}, ..., MK_{l1}^{(2)} >, < MK_1^{(1)}, MK_2^{(1)}, ..., MK_m^{(1)} >,$$
$$, < MK_1^{(2)}, MK_2^{(2)}, ..., MK_{l2}^{(2)} >>; i = \overline{1, M} \quad (18)$$

Тогда всю программу $\boldsymbol{PR}^{(j)}$ можно разделить на кортеж сегментов, определяемых как (18):

$$PR^{(j)} \rightarrow < SG_1^{(j)}, SG_2^{(j)}, ..., SG_m^{(j)} >; m = \overline{1, M}; \forall j = \overline{1, N}. \quad (19)$$

Под подпрограммой нейромикропрограммы RO_i будем понимать кортеж сегментов, определяемых как (18):

$$RO_l^{(j)} = < SG_1^{(j)}, SG_2^{(j)}, ..., SG_{ls}^{(j)} >; l = \overline{1, L}; \forall j = \overline{1, N} \quad (20)$$

На рисунке 1 показано сегментирование и программы $PR^{(j)}$.

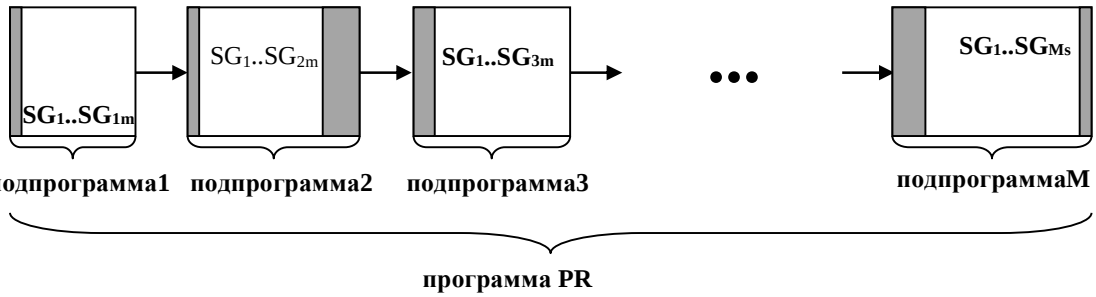


Рисунок 1 – Сегментирование программы $PR^{(j)}$

Рассмотрим две произвольно взятые подпрограммы RO_j и RO_k, каждая из которых – некоторый сегмент, то есть представляет собой множество команд эмуляции нейрона эквивалентных друг другу и вспомогательных команд, подготавливающих данные для обработки и являющихся неотъемлемой частью соответствующих команд первого типа:

$$\begin{aligned} &RO_n^{(j)} = < MK_1, MK_2, ..., MK_i, ..., MK_I >; MK_i = \{MK_i^{(1)}, MK_i^{(2)}\}; i = \overline{1, I} \\ &\forall MK_k^{(1)}, MK_l^{(1)} \in RO_n^{(j)} \in E : MK_k^{(1)} E MK_l^{(1)} \end{aligned}. \quad (21)$$

Рассмотрим отношение $RO_l S_w RO_k$ структуры S_w, которое указывает на то, что любые две произвольно взятые подпрограммы RO_j и RO_k, удовлетворяющие (20), могут выполняться одновременно на разных НПВМ под управлением микрокоманд, принадлежащих указанным подпрограммам, т.е.:

$$\forall RO_l, RO_k \in PR^{(j)} : RO_l S_w RO_k. \quad (22)$$

Тогда справедливо утверждение, что отношение структуры обработки S_w есть отношение эквивалентности.

Всякая программа обработки информации RO_l, удовлетворяющая (17), в связи с взаимной эквивалентностью атомарных операций подпрограммы может выполняться параллельно самой себе, т.е. $RO_l S_w RO_l$. Таким образом, справедливо условие рефлективности подпрограмм $\forall RO_l \in PR^{(j)}$.

Если подпрограмма обработки информации RO_l равна подпрограмме RO_k, то подпрограмма обработки RO_k равна подпрограмме обработки RO_l и одновременно с ней может выполняться, т.е.:

$$\forall RO_l, RO_k \in PR^{(j)} : RO_l S_w RO_k \rightarrow RO_k S_w RO_l. \quad (23)$$

Другими словами, выполняется условие симметричности двух подпрограмм обработки.

Если подпрограмма обработки информации RO_l равна подпрограмме обработки RO_k, а подпрограмма RO_k равна RO_q, тогда подпрограмма обработки RO_l равна и параллельна подпрограмме обработки RO_q, т.е.

$$\forall RO_l, RO_k \in PR^{(j)} \in S_w : RO_l S_w RO_k, RO_k S_w RO_q \rightarrow RO_l S_w RO_q . \quad (24)$$

Таким образом, справедливо условие транзитивности подпрограмм обработки информации.

Тогда задача разбиения программы на множество подпрограмм преобразуется в задачу рационального последовательного объединения сегментов $< SG_m >; m = \overline{1, M}$ в множество подпрограмм $\{RO_l^{(j)}\}; l = \overline{1, L}$:

$$PR^{(j)} \rightarrow < SG_1^{(j)}, SG_2^{(j)}, ..., SG_m^{(j)} > \rightarrow \{RO_l^{(j)}\}; m = \overline{1, M}; l = \overline{1, L}; \forall j = \overline{1, N} \quad (25)$$

Задача разбиения заключается в нахождении значения ls выражения (19) для всех подпрограмм $RO_l^{(j)}; l = \overline{1, L}$, где число подпрограмм равно числу НПВМ $L = q$:

$$\begin{cases} RO_l^{(j)} =< SG_1^{(j)}, SG_2^{(j)}, ..., SG_{ls}^{(j)} >; l = \overline{1, L}; \forall j = \overline{1, N} \\ L = q \end{cases} \quad (26)$$

Пусть число команд $MK^{(1)}$ в программе $PR^{(j)}$ определяемой как (19) равно $\left|MK_{PR}^{(1)}\right|$, число команд некоторого сегмента $SG_i^{(j)}$ равно $\left|MK_i^{(1)}\right|$.

Цель разбиения программы на подпрограммы состоит в достижении одинакового времени обработки для всех этапов, то есть, в данном случае, в достижении одинакового количества команд $MK^{(1)}$ для всех подпрограмм. Тогда необходимо определить среднее число $\overline{MK_{PR}^{(1)}}$ команд $MK^{(1)}$ в подпрограмме:

$$\overline{MK_{PR}^{(1)}} = \frac{\left|MK_{PR}^{(1)}\right|}{q} \quad (27)$$

Добавление нового сегмента $SG_i^{(j)}$ к подпрограмме возможно, когда разность значения $\overline{MK_{PR}^{(1)}}$ и суммарного количества команд $MK^{(1)}$, уже добавленных в подпрограмму $RO_l^{(j)}$ сегментов больше, чем $\frac{\left|MK_{PR}^{(1)}\right|}{2}$.

Если верно условие $\overline{MK_{PR}^{(1)}} - \sum_{ll=lb}^{le} \left|MK_l^{(1)}\right| \geq \frac{\left|MK_{PR}^{(1)}\right|}{2}$, (28)

где ll определяет номера сегментов, уже включенных в подпрограмму:

lb - номер первого сегмента в подпрограмме $RO_l^{(j)}$;

le - номер последнего сегмента в подпрограмме $RO_l^{(j)}$.

тогда $RO_l^{(j)} \to RO_l^{(j)} + SG_i^{(j)};\ le \to le+1$ (29)

Если условие (28) неверно, то

$RO_{l+1}^{(j)} \to RO_{l+1}^{(j)} + SG_i^{(j)}; lb = i; \ln = i$ (30)

Начальными условиями для решения задачи (26) являются: $lb = 1; \ln = 1$

Таким образом, в соответствии с выражениями (28)-(30), нейромикропрограмма $PR^{(j)}$ может быть рационально разделена на подпрограммы $PR^{(j)} \to \{RO_l^{(j)}\}; l = \overline{1, L}; \forall j = \overline{1, N}$, для которых верно отношение структуры обработки информации S_w.

4. Разработка методологических основ исследования связей между подпрограммами

Результаты идентификации связей между подпрограммами $\{RO_l^{(j)}\}; l = \overline{1, L}; \forall j = \overline{1, N}$ НПВС могут быть представлены в виде матрицы связи НПВМ $M = [M_{ij}]$. Ее размерность $q \times q$, где q - число НПВМ.

Каждый элемент матрицы может принимать значения: $M_{ij} = \{'-', 'n', 'y'\}$:

'n' (no) - нет связи между НПВМ;

'y' (yes) - есть связь между НПВМ (в подпрограмме $RO_k^{(j)}$ ресурсы подпрограммы $RO_i^{(j)}$ при том, что $k > i$);

'–' - между подпрограммами не может быть связи.

Для примера рассмотрена матрица связей для пяти подпрограмм $\{RO_l\}; l = \overline{1,5}$:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{X} & \boldsymbol{n} & \boldsymbol{n} & \boldsymbol{n} & \boldsymbol{n} \\ \boldsymbol{X} & \boldsymbol{X} & \boldsymbol{y} & \boldsymbol{y} & \boldsymbol{n} \\ \boldsymbol{X} & \boldsymbol{X} & \boldsymbol{X} & \boldsymbol{n} & \boldsymbol{n} \\ \boldsymbol{X} & \boldsymbol{X} & \boldsymbol{X} & \boldsymbol{X} & \boldsymbol{y} \\ \boldsymbol{X} & \boldsymbol{X} & \boldsymbol{X} & \boldsymbol{X} & \boldsymbol{X} \end{bmatrix}.$$

Для определения элементов матрицы M вводится вспомогательная матрица $M' = [M_{ij}^{'}]$ размерности $q \times E$, в которой число столбцов E – это количество аппаратных ресурсов процессора (регистры, ячейки памяти и т.п.). Например, для нейропроцессоров семейства NM640x $E = 44$ (дополнительно заносятся области памяти, занятые переменными).

Элементами матрицы могут быть $M'_{ij} = \{'-', 'g', 's', 'gs'\}$:

'–' - элемент не использовался в подпрограмме $RO_i^{(j)}$;

's' (set) - элемент был присвоен в подпрограмме $RO_i^{(j)}$;

'g' (get) - элемент был использован в подпрограмме $RO_i^{(j)}$;

'gs' (get+set) - элемент был использован, а затем присвоен в одной и той же подпрограмме $\{RO_i^{(j)}\}$.

Для примера рассмотрена матрица M' для пяти подпрограмм $\{RO_l\}; l = \overline{1,5}$:

$$M' = \begin{bmatrix} - & - & - & ... & - \\ s & - & s & ... & - \\ - & g & - & ... & - \\ - & s & - & ... & s \\ gs & - & - & ... & - \end{bmatrix}.$$

В статье не рассматривается вариант, когда ресурсу сначала присваивается значение в подпрограмме, а затем используется, т.к. важна только ситуация присвоения элемента в текущей подпрограмме, т.е. случай с обозначением 's'.

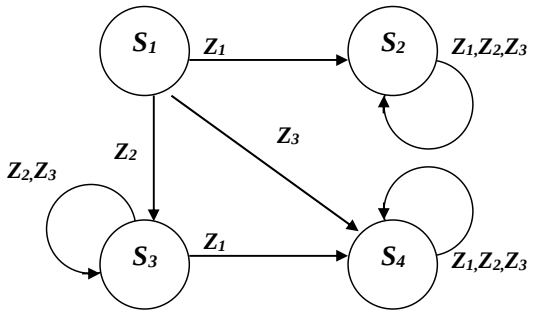
Данная матрица отражает состояние каждого элемента процессора во всех подпрограммах $RO_i^{(j)}$ и позволяет определить элемент матрицы M_{ij} следующим образом:

$$((M'_{ik} = 's') \wedge (M'_{jk} = 'g') \vee (M'_{ik} = 'gs')) \wedge ((M'_{nk} = 's') \vee (M'_{nk} = 'gs')) \rightarrow$$
$$\rightarrow M_{ij} = 'y'; n = \overline{i+1, j}; \quad (31)$$

$$((M'_{ik} = 'gs') \wedge (M'_{jk} = 'g') \vee (M'_{ik} = 'gs')) \wedge ((M'_{nk} = 's') \vee (M'_{nk} = 'gs')) \rightarrow$$
$$\rightarrow M_{ij} = 'y'; n = \overline{i+1, j}. \quad (32)$$

Целью алгоритма исследования и идентификации связей является заполнение элементов матрицы связей НПВМ.

Для определения состояния ресурса процессора R в подпрограмме $RO_i^{(j)} = P$ предлагается использовать математический аппарат конечных автоматов. Пусть $K = (S, Z, W, \delta, \lambda, S_1)$ - конечный автомат описания (рисунок 2).



$$\delta = \begin{bmatrix} \delta & S_1 & S_2 & S_3 & S_4 \\ Z_1 & S_2 & S_2 & S_4 & S_4 \\ Z_2 & S_3 & S_2 & S_3 & S_4 \\ Z_3 & S_4 & S_2 & S_3 & S_4 \end{bmatrix}$$

Рисунок 2 – Представление конечного автомата K идентификации состояния аппаратного ресурса НПВМ R

Множество состояний S автомата K:

S_1 - состояние, определяющее значение $M'_{RP} = '-'$;

S_2 - состояние, определяющее значение $M'_{RP} = 's'$;

S_3 - состояние, определяющее значение $M'_{RP} = 'g'$;

S_4 - состояние, определяющее значение $M'_{RP} = 'gs'$;

Входной алфавит Z автомата K:

Z_1 - ресурс R поменял значение;

Z_2 - значение ресурса R было использовано;

Z_3 - значение элемента R было сначала использовано, потом присвоено.

Выходной алфавит W автомата K:

$W_1 = \lambda(S_1)$; $W_2 = \lambda(S_2)$; $W_3 = \lambda(S_3)$; $W_4 = \lambda(S_4)$.

Таким образом, результатом реализации этапа является некоторая матрица $M = [M_{ij}]$, описывающая зависимости между подпрограммами $\{RO_l^{(j)}\}; l = \overline{1, L}; \forall j = \overline{1, N}$.

4. Классификация и выбор структур НПВС

Рассмотрим введенное ранее понятие структуры S_w, под которым понимается отношение параллельности выполнения подпрограмм RO_{i1}, RO_{i2} между i_1-м и i_2-м процессорными модулями $R_{i1} S_w R_{i2}; \forall R_{i1}, R_{i2} \in PR^{(j)}$. Тогда необходимо определить структуру $S_w \in S; w = \overline{1, W}$ из множества всевозможных структур $S = \{S_1, S_2, ..., S_w ..., S_W\}$, позволившую некоторой j-й программе $PR^{(j)}$, определяемую как (11) поставить в соответствие множество подпрограмм $\{RO_l^{(j)}\}; l = \overline{1, L}$ [13]. Эта структура описывается типом, числом процессорных модулей q и матрицей связей между ними [14].

Если рассмотреть все возможные структуры, полученные в результате решения задачи (13), то могут получиться уже известные структуры обработки: конвейерная, векторная, конвейерно-векторная, векторно-конвейерная и множество неизвестных структур, которые можно классифицировать путем введения лингвистических переменных, например, почти-векторная, почти-конвейерная и т.п.

1. Выходная информация каждой подпрограммы RO_i является входной для следующей $RO_{i+1}; i = 1, q$; входная первой программы RO_1 является входной, а последней RO_q – выходной всего устройства, тогда назначая q процессорных модулей на обработку информации, имеем структуру конвейерного типа. Отличие матрицы для данной структуры – наличие значений $'y'$ по диагонали над главной диагональю матрицы. В остальных ячейках должны быть значения $'n'$. Пример матрицы конвейерной структуры для пяти НПВМ:

$$M = \begin{bmatrix} - & y & n & n & n \\ - & - & y & n & n \\ - & - & - & y & n \\ - & - & - & - & y \\ - & - & - & - & - \end{bmatrix}$$

Структура имеет смысл при распределении слоев нейронной сети на разные НПВМ, то есть «вертикальное» разделение искусственной нейронной сети (ИНС).

2. Входная информация требуется одновременно для всех подпрограмм, тогда назначая q процессорных модулей на обработку информации, получаем структуру векторного типа, в которой все q НПВМ будут функционировать одновременно.

Отличием матрицы для данной структуры является то, что все значения ячеек равны $'n'$. Пример матрицы векторной структуры для НПВМ:

$$M = \begin{bmatrix} - & n & n & n & n \\ - & - & n & n & n \\ - & - & - & n & n \\ - & - & - & - & n \\ - & - & - & - & - \end{bmatrix}$$

Структура имеет смысл при разделении нейронной сети на сегменты нейронов, которые могут выполняться на разных НПВМ, то есть «горизонтальное» разделение ИНС.

3. При объединении смежных подпрограмм, в которых информация передается параллельно, в K множеств: $G_i = RO_{iM}^{(j)} \cup ... \cup RO_{iN}^{(j)}, \forall i = \overline{1,K}$, – тогда $G = \{G_1,...,G_K\}$. Число подпрограмм в каждой группе различно для каждого множества $N_{G_i} = |\{G_i\}|, \forall i = \overline{1,K}$. Тогда, если все подпрограммы $RO_{iM}^{(j)},...,RO_{iN}^{(j)}$ внутри множеств $G_i, i = \overline{1,K}$, обмениваются информацией последовательно, то структура является конвейерно-векторной, в которой по принципу конвейерной системы обрабатываются векторные структурные блоки.

Число НПВМ будет равно $q = \sum_{i=1}^{K} |\{G_i\}|$. (33)

Отличием матрицы для данной структуры является то, что все значения $'y'$ сгруппированы в прямоугольные контуры, которые упорядочены в матрицы по типу лестницы и нет рядов и строк матрицы без значения $'y'$. Пример матрицы векторной структуры для пяти НПВМ:

$$M = \begin{bmatrix} - & y & y & y & n \\ - & - & n & n & y \\ - & - & - & n & y \\ - & - & - & - & y \\ - & - & - & - & - \end{bmatrix}$$

Структура имеет смысл при разделении общей нейронной сети на комплекс самостоятельных ИНС, которые могут выполняться на разных НПВМ.

Если в каждом множестве G_i число подпрограмм равно 1, т.е. $|G_1| = |G_2| = ... = |G_K| = 1$, то структура конвейерного типа.

4. При объединении всех смежных подпрограмм, в которых информация передается последовательно, в V множеств: $G_i = RO_{iM}^{(j)} \cup ... \cup RO_{iN}^{(j)}, \forall i = \overline{1,V}$, – тогда $G = \{G_1, ..., G_V\}$. Число подпрограмм в каждой группе различно для каждого множества $N_{G_i} = |\{G_i\}|, \forall i = \overline{1,V}$. Если всем подпрограммам $RO_{iM}^{(j)}, ..., RO_{iN}^{(j)}$ внутри множества $G_i, i = \overline{1,V}$ информация необходима одновременно, то структура является векторно-конвейерной, в которой по векторному принципу обрабатываются конвейерные структурные блоки.

Число НПВМ будет равно $q = \sum_{i=1}^{V} |\{G_i\}|$. (34)

Отличием матрицы структуры является то, что все значения ячеек равны 'n', кроме некоторых (не всех) значений 'y' над главной диагональю. Пример матрицы векторно-конвейерной структуры для пяти НПВМ:

$$M = \begin{bmatrix} - & y & n & n & n \\ - & - & n & n & n \\ - & - & - & y & n \\ - & - & - & - & n \\ - & - & - & - & - \end{bmatrix}$$

Структура имеет смысл при резком увеличении количества нейронов ИНС на следующих слоях сети, в этом случае рационально разделить ИНС по векторному принципу, но на некоторых слоях.

Если в каждом множестве G_i число подпрограмм равно 1, т.е. $|G_1| = |G_2| = ... = |G_V| = 1$, то структура векторного типа.

5. Во всех остальных случаях (например, если группы обмениваются информацией последовательно, часть НПВМ внутри группы – параллельно, часть – последовательно) имеем НПВС произвольной структуры.

Таким образом, рассмотрены варианты возможных многопроцессорных структур и определено количество процессорных модулей для каждого типа структуры.

Эксперты выделяют многопроцессорные системы с «сильной» связью, которые содержат несколько процессоров, подключаемые на шинном уровне и многопроцессорные системы с «гибкой» связью, основанные на множественных автономных одиночных или двойных компьютерах, объединяемых через высокоскоростные каналы передачи данных, таких как Gigabit Ethernet [2].

Классификация вычислительных систем на базе нейропроцессоров, исходя из вида связей между ее элементами:

1. Параллельные нейропроцессорные вычислительные системы, элементы которых соединены на шинном уровне [8].

Параллельная система может быть описана в виде:

$$RS = \{P, E, \Delta, S_w\}, \quad (35)$$

где $P = \{P_1, P_2, ..., P_q\}$ – множество вычислительных узлов РНВПС, где каждый вычислительный узел может быть описан с помощью технических параметров:

$$P_i \to H_i, H_j, H_k, H_m, \quad (36)$$

где H_i, H_j, H_k – технические параметры, отвечающие за тактовую частоту нейропроцессора, количество ядер, объемы оперативной и дисковой памяти соответственно;

S_w – структура РНВПС, выбор которой описан ранее

E – множество направленных связей между узлами НПВС, количество которых определяется видом структуры, то есть $E = F(S_w)$;

Δ – диспетчер (управляющий узел) НПВС.

Так как процессоры соединены между собой на шинном уровне, то задержками передачи информацией Ttr можно пренебречь: $Ttr \approx 0$.

Пример параллельной вычислительной структуры показан на рисунке 3.

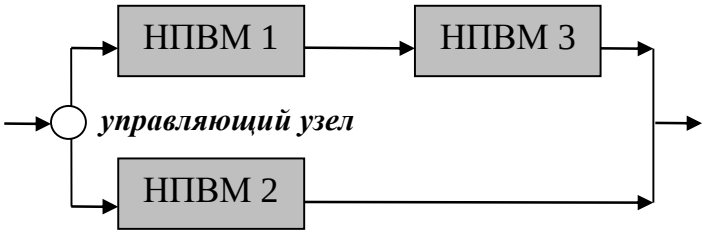


Рисунок 3 – Структура с «сильной» связью

2. Распределенные нейропроцессорные вычислительные системы (РНПВС), представляющие собой совокупность НПВМ или самостоятельных вычислительных систем расположенных друг от друга на значительном расстоянии, но объединенных с помощью программно управляемых коммутаторов и системных устройств. Следует отметить, что понятия параллельной и распределенной обработки не являются эквивалентными. Частным случаем распределенной системы является GRID-система [9]. Пример функционирования РНПВС показан на рисунке 4.

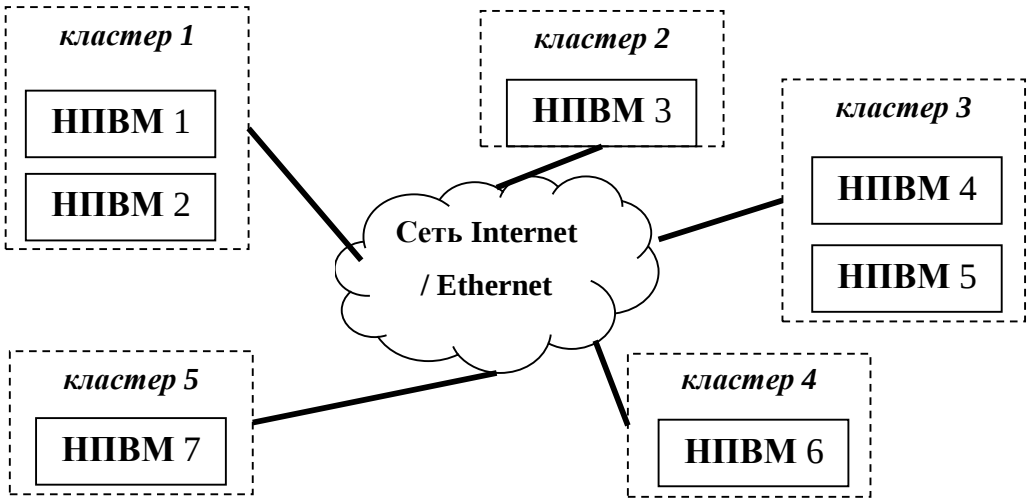


Рисунок 4 – Функционирование распределенной системы

Одним из свойств распределенных систем является отсутствие общей памяти, кроме случая предоставления абстракции единого адресного пространства [7], который в работе не рассматривается.

Распределенная система может быть описана в виде:

$$RS = \{CL, E, \Delta, S_w, Ttr, Tl\}, \quad (37)$$

где $CL = \{CL_1, CL_1 ..., CL_r\}$ – множество вычислительных кластеров РНПВС;

E – множество направленных связей между кластерами РНПВС;

Δ – диспетчер (управляющий узел) РНПВС;

S_w – структура РНПВС;

Ttr – средняя пропускная способность между узлами РНПВС;

Tl – латентность, то есть временные затраты, необходимые для инициализации сообщений, отправления и приема данных и т.п.

Каждый кластер CL_i может быть описан следующим образом:

$$CL_i = \{P_i, E_i\}, \quad (38)$$

где $P = \{P_{i1}, P_{i2}, ..., P_{ig}\}$ - множество вычислительных узлов кластера CL_i;

E_i – множество связей между ними.

Целесообразно представить РНПВС в общем виде без учета кластеризации.

Тогда, распределенная система может быть описана следующим образом:

$$RS = \{P_i, E, \Delta, S_w, Ttr, Tl\}. \quad (39)$$

где $P = \{P_1, P_2, ..., P_q\}$ – множество вычислительных узлов РНВПС;

E_i – множество связей между ними, притом, что некоторые связи могут быть «сильными», а некоторые «слабыми»;

Рассмотрены множества, определяющие временные задержки передачи данных и латентности при передаче данных Ttr и Tl для РНПВС.

$Ttr = \{Ttr_{ij}\}$, где

Ttr_{ij} - время передачи данных между i и j НПВМ

Время передачи данных может быть:

$$\begin{cases} 0 & \text{, если связь в кластере} \\ Ttr_{ij} = Ttr_{i0} + Ttr_{j0} & \text{, в противном случае,} \end{cases} \quad (40)$$

где Ttr_{i0} - время передачи данных от i-го узла к управляющему узлу Δ.

$$Ttr_{ij} = PS * N, \quad (41)$$

где PS - пропускная способность канала передачи данных (в бит/с);

N - количество бит для передачи.

$$Ttr = \sum_{i=1}^{|E|} Ttr_{ij}, \quad (42)$$

$$Tl = \{Tl_{ij}\}, \quad (43)$$

где Tl_{ij} - время латентности при передаче данных между i и j НПВМ

Время латентности может быть:

$$\begin{cases} 0 & \text{, если связь в кластере} \\ Tl_{ij} = Tl_{i0} + Tl_{0j} & \text{, в противном случае,} \end{cases} \quad (44)$$

где Tl_{i0} - время латентности при передаче данных от i-го узла к управляющему узлу Δ;

Tl_{0j} - время латентности при передаче данных от управляющего узла Δ к j-му узлу.

$$Tl = \sum_{i=1}^{|E|} Tl_{ij}. \quad (45)$$

Пример структуры РНПВС показан на рисунке 5.

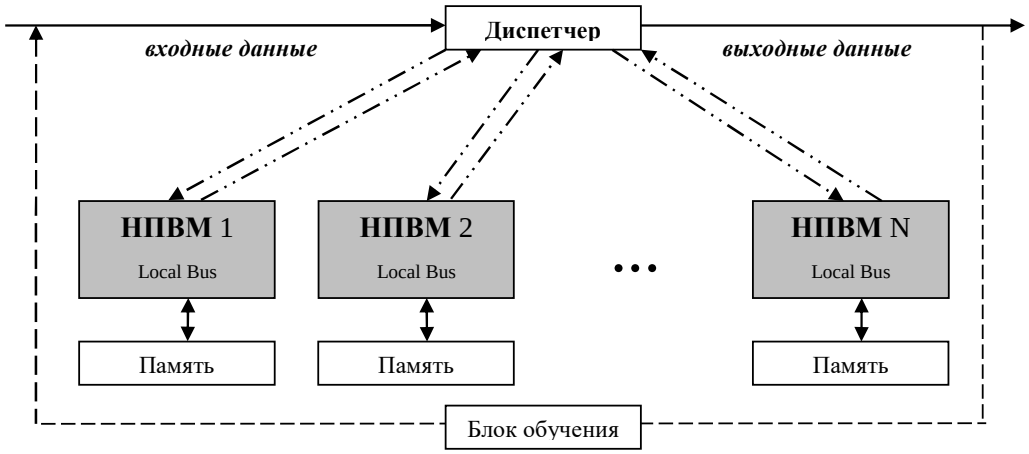


Рисунок 5 – Распределенная структура

3. Облачные нейропроцессорные вычислительные системы (ОНПВС), предполагающие обеспечение повсеместного сетевого доступа по требованию к инфраструктуре конфигурируемых вычислительных ресурсов, таких как

нейропроцессорная система (рассматривается только модель обслуживания IaaS – предоставление пользователям нейрокомпьютерной инфраструктуры). Принципиальным отличием облачной системы от параллельных и распределенных является наличие временных задержек за счет использования относительно медленных каналов связи между пользователем и «облаком» [8].

На рисунке 6 показан пример функционирования облачной системы.

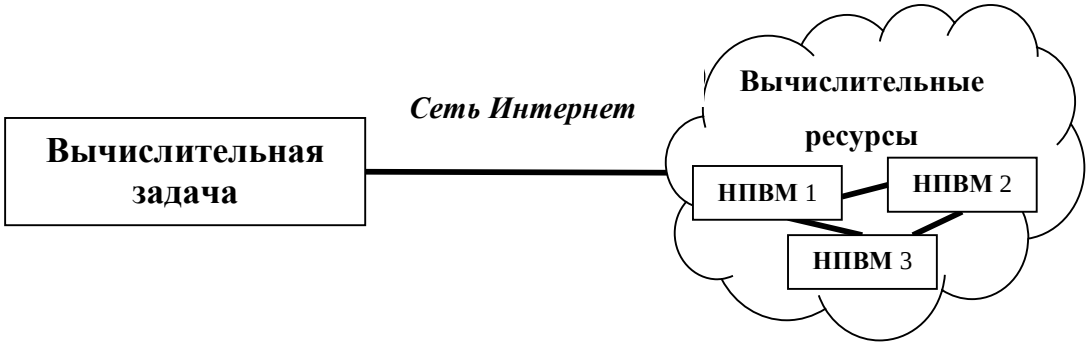


Рисунок 6 – Функционирование облачной системы

В случае облачной системы возможны следующие два варианта:

а) вычислительная система в «облаке» не имеет распределенной структуры, (рис. 7а).

Такая система может быть описана так же, как и РНВПС (39).

В этом случае величина $Ttr \approx 0$, а время задержек $Ttrc$ в системе равно:

$$Ttrc = Ttr_{po} + Ttr_{op}, \quad (46)$$

где Ttr_{po} – время передачи данных между пользователем и узлом Δ;

Ttr_{op} - время передачи данных между узлом Δ и пользователем.

$$Tlc = Tl_{po} + Tl_{op}, \quad (47)$$

где Tl_{po} – время латентности при передаче данных между пользователем и Δ;

Tl_{op} – время латентности при передаче данных между Δ и пользователем.

б) Вычислительная система в «облаке» имеет распределенную структуру (рис. 7б).

Такая система может быть описана следующим образом:

$$RS = \{P_i, E, \Delta, S_w, Ttr, Tl, Ttrc, Tlc\}, \quad (48)$$

где $Ttrc$ – время задержек передачи данных до управляющего устройства НПВС и обратно к пользователю;

Tlc – время латентности при передаче данных от управляющего устройства НПВС и обратно.

В этом случае общее время временных издержек Ts передачи данных определяется как сумма (42),(47),(45),(48):

$$Ts = \sum_{i=1}^{|E|} Tl_{ij} + Ttr_{po} + Ttr_{op} + \sum_{i=1}^{|E|} Tl_{ij} + Tl_{po} + Tl_{op} \quad (49)$$

$$Ts = Ttrc + Tlc + Ttr + Tl \neq 0 \quad (50)$$

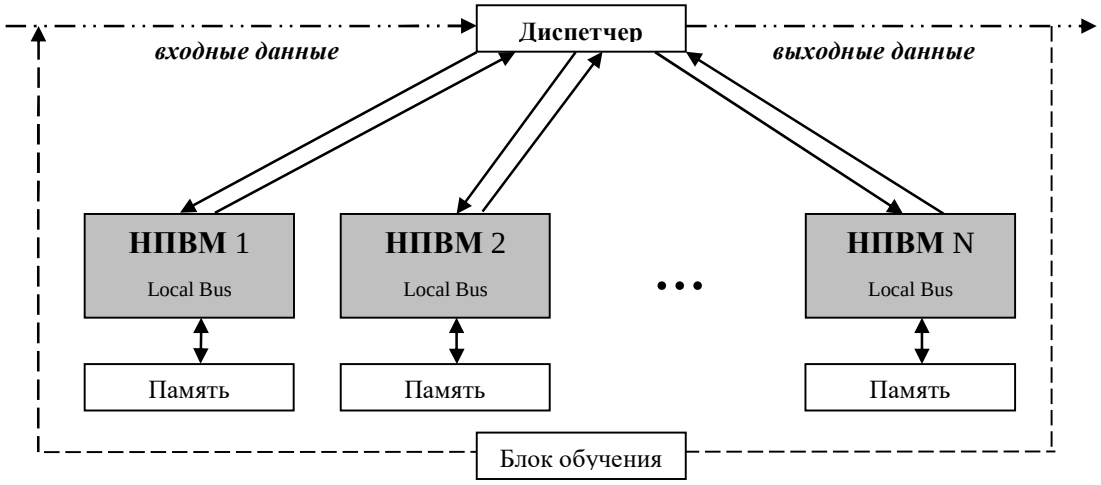


Рисунок 7а – Облачная система, не имеющая распределенной структуры

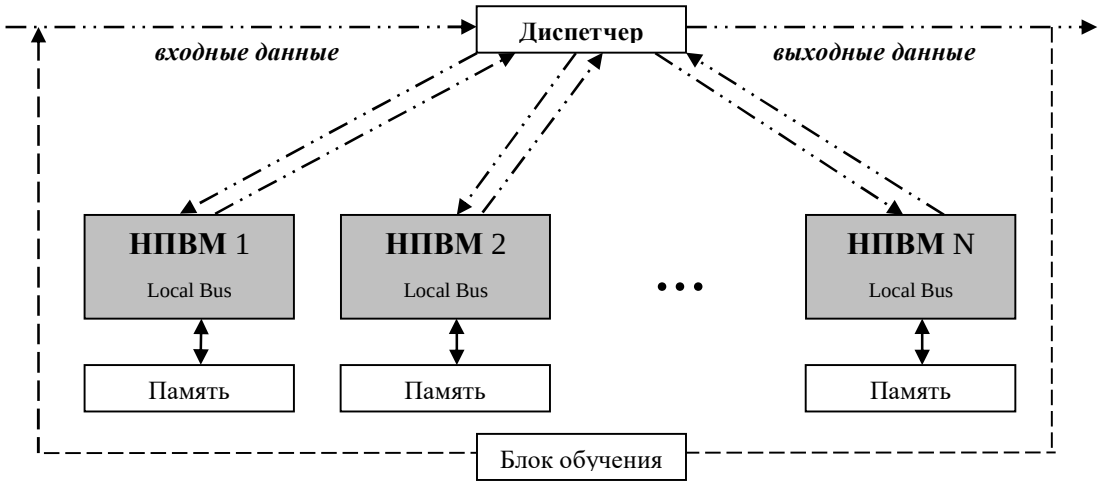


Рисунок 7б – Облачная система, имеющая распределенную структуру

5 Программные средства организации вычислительных систем параллельной и распределенной обработки данных на базе нейропроцессоров

В качестве программных средств организации вычислительных систем параллельной и распределенной обработки данных на базе нейропроцессоров разработана модульная программная платформа «NP Studio» [10], состоящая из следующих подсистем.

5.1. Специализированный текстовый редактор для языков нейроассемблера

В подсистеме (рисунок 8) реализованы задачи создания и разбиения программного кода на языках нейроассемблера PR_{ASM}, специального языка C++ PR_{C++}, компилируемого программным обеспечением НТЦ «Модуль», файлов макросов PR_{ASM}.

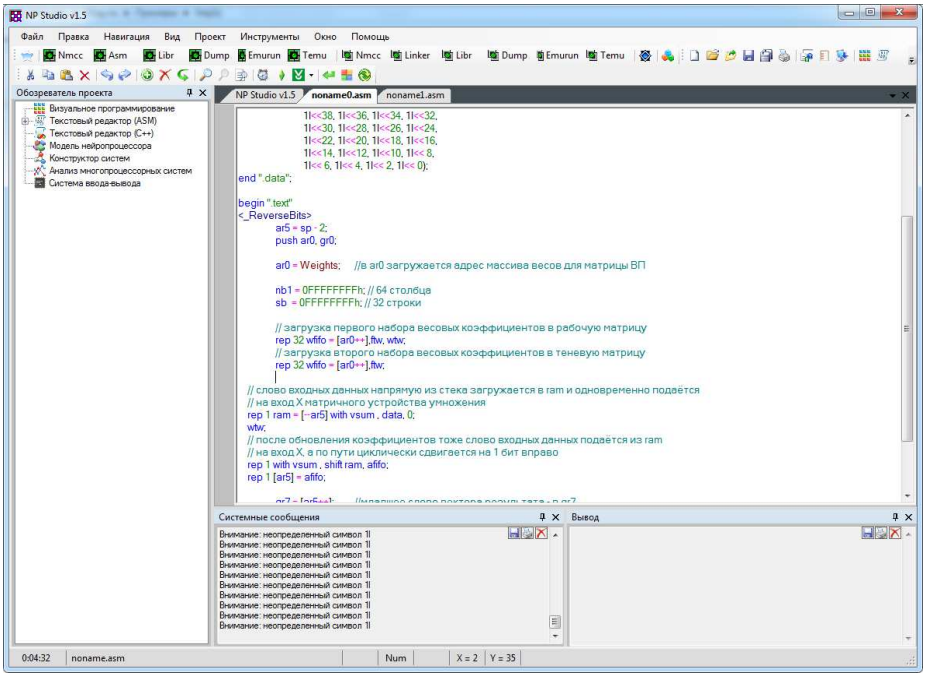


Рисунок 8 – Форма текстового редактора

По результатам анализа программы PR_{ASM} возможна генерация матрицы связей M исходя из алгоритма задачи (рисунок 9).

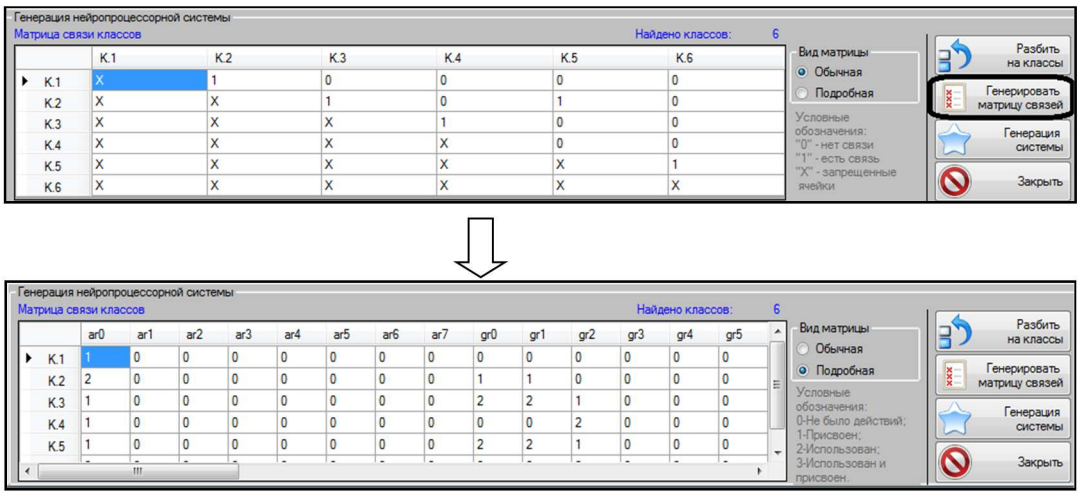


Рисунок 9 – Генерация матрицы связей подпрограмм

5.2. Конструктор систем

В данной части реализован блок выбора используемой системы S_w из множества структур $S_w \in S; w = \overline{1,W}$. В соответствии с классификацией структур, описанной ранее, должен быть реализован выбор из пяти вариантов: векторная, конвейерная, векторно-конвейерная, конвейерно-векторная и произвольная архитектура. Кроме этого важен тип связи: «сильная», «слабая».

Форма конструктора систем для векторно-конвейерной структуры на базе эмулятора процессора NM6406 приведена на рисунке 10.

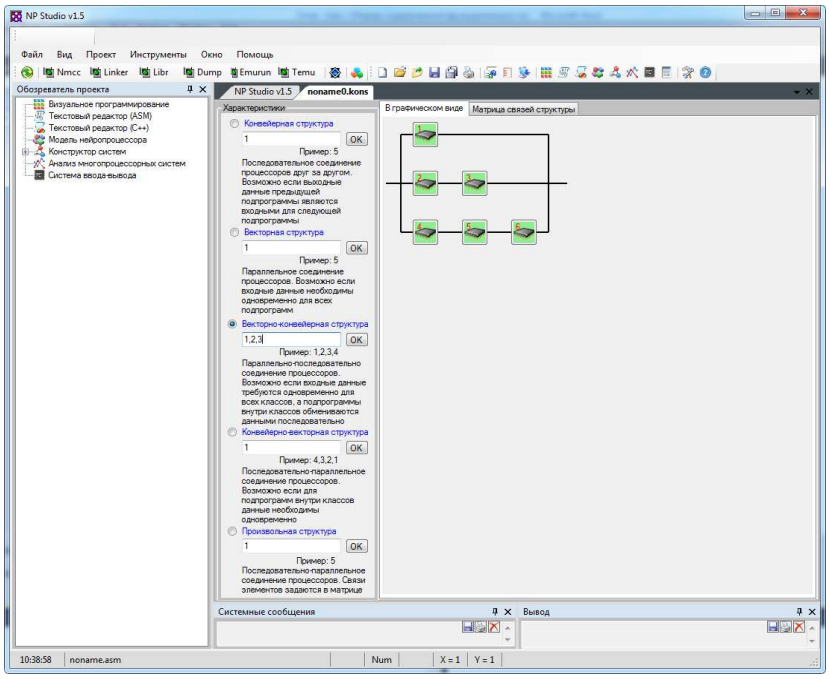


Рисунок 10 – Форма конструктора систем

Выбор структуры происходит в подсистеме «Текстовые редактор» (рисунок 11), представление которой можно видеть в подсистеме «Конструктор систем».

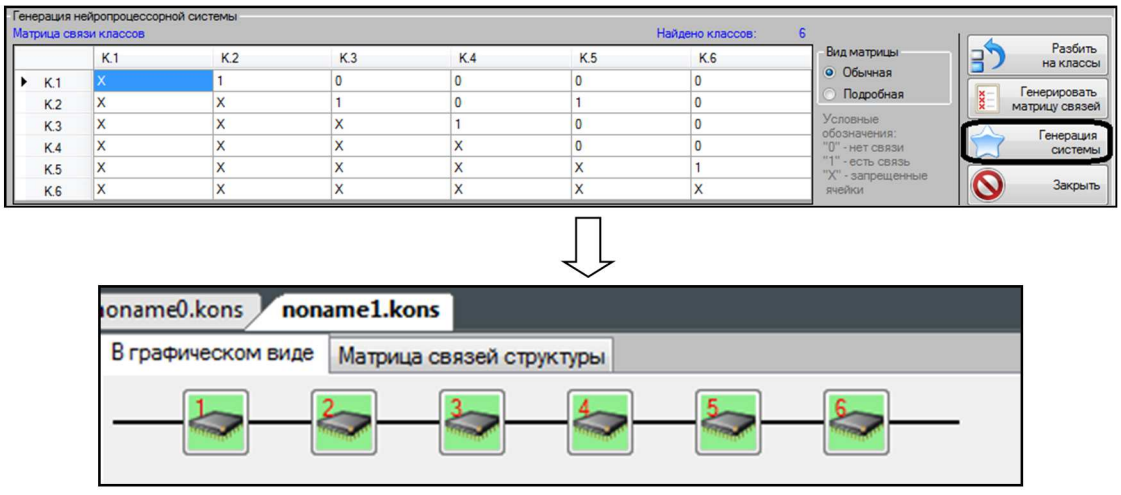


Рисунок 11 – Генерация НПВС на основе матрицы связей

Разработанные программные средства использованы для реализации проекта «Разработка программного обеспечения нейропроцессорной системы автоматического управления гексаподом» совместно с Институтом проблем машиноведения Российской академии наук (договор № 5 от 1 октября 2014 г.).

Заключение

Результатами исследования являются:

- теоретико-множественная модель организации вычислительных систем параллельной и распределенной обработки данных на базе нейропроцессоров;
- метод рационального разбиения нейромикропрограммы на множество подпрограмм,
- метод исследования и идентификации связей между нейроподпрограммами,
- методика рационального выбора нейропроцессорной структуры;
- классификация нейропроцессорных структур по двум критериям: зависимость между подпрограммами и вид связей между нейропроцессорами;
- программные средства организации вычислительных систем параллельной и распределенной обработки данных на базе нейропроцессоров.

Таким образом, предлагается решение для проектирования и логической организации параллельных, распределенных и облачных нейропроцессорных систем, кроме этого результаты могут быть использованы в дальнейшем для решения задачи анализа многопроцессорных систем на базе нейропроцессоров и их оптимизации.

Список информационных источников

[1] Vladimir Ruchkin, Vitaliy Romanchuk, Roman Sulitsa. Clustering, Restorability and Designing Of Embedded Computer System Based On Neuroprocessors // Proceedings of the 2nd Mediterranean Conference on Embedded Computing (MECO). – Budva, Montenegro, 2013. – P. 58-62.

[2] Букатов, А.А., Дацюк, В.Н., Жегуло, А.И. Программирование многопроцессорных вычислительных систем. – Ростов-на-Дону: Издательство ООО «ЦВВР», 2003. – 208 с.

[3] Галушкин, А.И. Нейрокомпьютеры: Кн.3. – М: ИПРЖР, 2000. – 524 с

[4] Горбань, А.Н. Нейроинформатика: кто мы, куда мы идем, как путь наш измерить // Вычислительные технологии. – 2000. – № 4. – С. 10-14.

[5] Долинский, М.С., Толкачев, А.А. Обзор аппаратных и программных средств реализации параллельной обработки // Компоненты и технологии. – 2004. – №6. – С. 54-56.

[6] Злобин, В.К., Григоренко, Д.В., Ручкин, В.Н., Романчук, В.А. Кластеризация и восстанавливаемость нейропроцессорных систем обработки данных // Известия тульского государственного университета. Технические науки. – 2013. – Вып.9.: Ч.2. – С.125-135.

[7] Комарцова, Л.Г., Максимов, А.В. Нейрокомпьютеры. – М.: Изд-во МГТУ им. Н.Э.Баумана, 2004. – 400 с.
[8] Полежаев, П.Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Параллельные вычислительные технологии (ПАВТ'2010). – Челябинск: ЮУрГУ, 2010. – С. 287–298.
[9] Полежаев, П.Н. Экспериментальное исследование алгоритмов планирования задач для вычислительной грид-системы // Системы управления и информационные технологии. – 2011. – №3.2(45). – С. 266–270.
[10] Романчук, В.А. Инновационный программный комплекс моделирования вычислительных систем на базе нейропроцессоров «НейроКС» // Современные научные исследования и инновации. – Декабрь, 2012 [Электронный ресурс]. – URL: http://web.snauka.ru/issues/2012/12/19407.
[11] Романчук, В.А. Моделирование нейропроцессорных систем // Отраслевые аспекты технических наук. – 2013. – №10(34). – С.19–24.
[12] Романчук, В.А. Разработка алгоритмов определения связей элементов вычислительной структуры на базе нейропроцессоров // Информатика и прикладная математика. – Рязань: РГУ имени С.А. Есенина, 2011. – Вып.17. – С.102–105.
[13] Романчук, В.А., Ручкин, В.Н. Алгоритмы анализа вычислительных структур на базе нейропроцессоров // Вестник РГРТУ. –2012. – №2. (Вып.40.) – С. 60–66.
[14] Романчук, В.А., Ручкин, В.Н. Разработка алгоритмов определения вида структуры нейропроцессорной системы на основе описания связей ее элементов // Информатика и прикладная математика: межвуз. сб. науч. тр. – Рязань: РГУ имени С.А. Есенина, 2011. – Вып.17. – С.106–109.
[15] Романчук, В.А., Ручкин, В.Н., Фулин, В.Н. Проектирование нейропроцессорных систем на основе нечеткой кластеризации // Вестник РГРТУ. –№4. (Вып.50-1). – С.87–93.
[16] Ручкин, В.Н., Романчук, В.А., Фулин, В.А. и др. Экспертная система нечеткой кластеризации нейропроцессорных систем // Известия Тульского государственного университета. Технические науки. – 2014. – Вып.6. – С. 162-167.