УДК 004.41[2+4+5]::004.272::004.855

# УВЕЛИЧЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ПРОГРАММНОЙ ПЛАТФОРМЫ ДЛЯ РЕАЛИЗАЦИИ АЛГОРИТМОВ МЕТОДА ГРУППОВОГО УЧЕТА АРГУМЕНТОВ

А.А. Орлов

Томский Государственный Университет Систем Управления и Радиоэлектроники
E-mail: d1scnc@gmail.com

*В предыдущих работах автором была предложена универсальная программная платформа, позволяющая реализовать известные алгоритмы метода группового учета аргументов, базисы, методы обучения и критерии селекции моделей. В статье приводится решение актуальной задачи увеличения производительности этой платформы. На основе проведенного обзора существующих архитектур вычислительных систем для параллельной обработки данных и программных систем индуктивного моделирования с поддержкой параллельных вычислений выработаны требования к подсистемам параллельных вычислений и управления памятью программной платформы. С использованием методологии объектно-ориентированного анализа и проектирования разработана объектно-ориентированная структура этих подсистем, приведены особенности их работы для каждой из указанных архитектур вычислительных систем. Производительность параллельной реализации комбинаторного алгоритма метода группового учета аргументов на базе программной платформы оценена экспериментально для многоядерных процессоров.*

***Ключевые слова:***
*Программная платформа, объектно-ориентированный анализ и проектирование, параллельная обработка данных, эффективность распараллеливания, метод группового учета аргументов.*

## Введение

Метод группового учета аргументов (МГУА) является методом структурно-параметрической идентификации сложных объектов, процессов и систем по данным наблюдений в условиях неполноты информации [1]. МГУА показал свою эффективность в самых различных областях: распознавание образов, нахождение физических и нефизических закономерностей, идентификация нелинейных систем, краткосрочное и долгосрочное прогнозирование стационарных и нестационарных процессов, управление сложными техническими объектами и в др. приложениях [2–4]. Столь широкое применение МГУА обусловило разработку множества алгоритмов метода, относящихся к различным типам: параметрических (комбинаторных и итерационных [5]), непараметрических [6. С. 103–124] и многорядных [7]. При этом зачастую реализация алгоритмов сопровождалась разработкой программного обеспечения «с нуля», как то Knowledge Miner [6. С. 147–176], FAKE GAME [8], GMDH Shell [9], ППП МГУА [10], GEvoM [11] и т. д. В работе [12] автором была показана актуальность создания единой программной платформы (далее по тексту – «Платформа МГУА»), позволяющей реализовать все известные алгоритмы МГУА, были выработаны требования к архитектуре (гибкости, универсальности и производительности), и предложена объектно-ориентированная структура такой программной платформы. Данная статья посвящена решению задачи увеличения производительности предложенной программной платформы МГУА.

## Анализ механизмов распараллеливания вычислений в существующих программных системах МГУА и аппаратных архитектурах с целью выработки требований к увеличению производительности единой программной платформы

МГУА, являясь индуктивным методом [1], осуществляет построение модели по некоторой выборке данных путем перебора структур моделей различной сложности из некоторого заданного множества. В связи с этим все алгоритмы МГУА, несмотря на их разнообразие [2, 13], начинают работу с генерации моделей на основе заданного множества структур. Параметрические алгоритмы МГУА следующим шагом осуществляют расчет параметров сгенерированных моделей таким образом, чтобы для каждой модели максимизировать значение заданного критерия качества («обучение моделей» – решение задачи параметрической оптимизации). После этого производится оценка каждой модели с использованием заданного внешнего критерия селекции моделей, причем, как правило, при этом используется часть выборки данных, не использовавшаяся при обучении моделей («расчет критерия»). Выбор модели оптимальной сложности («селекция моделей») осуществляется на основании значения критерия для каждой модели.

Как правило, качество результирующей модели с точки зрения внешнего критерия получается тем выше, чем большее количество моделей-кандидатов было обработано в процессе работы алгоритма МГУА. Увеличение количества моделей, в свою очередь, приводит к соответствующему уве-

личению времени работы алгоритмов МГУА, являющихся по своей сути переборными (например, для построения полиномиальной модели максимальной степени 2 для выборки данных из трех входных аргументов и одной выходной величины с помощью комбинаторного алгоритма МГУА требуется осуществить генерацию, обучение и применение критерия для 1024 моделей; при максимальной степени 3 – $3 \cdot 10^6$ моделей; 4 – $3 \cdot 10^{10}$ моделей [14]). Таким образом, качество результирующей модели тем выше, чем большее количество моделей обрабатывается в единицу времени, то есть чем более производительной является реализация алгоритма в рамках платформы МГУА.

Одним из способов повышения производительности является уменьшение времени, затрачиваемого на осуществление этапов генерации, обучения и применения критерия к каждой модели. Это возможно как путем алгоритмической оптимизации, так и используя различные приемы программной оптимизации [15, 16], применение которых специфично для каждой конкретной реализации каждого алгоритма в рамках единой платформы МГУА. С другой стороны, алгоритмы МГУА производят обучение и расчет значения критерия для каждой модели независимо от других моделей. Это позволяет проводить данные операции *параллельно* для всех моделей, кратно увеличивая производительность расчетов. В этом случае организация параллельных вычислений является ответственностью самой платформы МГУА.

В табл. 1 приведены программные системы индуктивного моделирования, реализующие алгоритмы МГУА и поддерживающие параллельные вычисления. Для каждой из систем приведено краткое описание способа распараллеливания – для всех систем характерен подход, основанный на выделении групп моделей, обрабатываемых независимо. Также приведены параметры экспериментов и практически полученный прирост производительности (рассчитываемый как отношение $\tau_P = t_1/t_P$ времени вычислений без распараллеливания $t_1$ и с распараллеливанием $t_P$). Из табл. 1 видно, что независимо от конкретных особенностей алгоритма МГУА и выборки данных эффективность распараллеливания ($\eta_P = \tau_P/P$, где $P$ – число параллельных потоков вычислений) может достигать величин выше 0,9.

В правой части табл. 1 указаны известные на сегодняшний день архитектуры вычислительных систем для параллельной обработки данных: от потоковых процессоров до распределенных систем. Плюсом отмечены ячейки тех архитектур, для которых описана поддержка со стороны соответствующей системы моделирования. Как видно из таблицы, все системы ориентированы на использование МЯП и МПС в качестве основной архитектуры, а Parallel Combi обладает поддержкой КЛ.

Рассмотрим подробнее упомянутые в табл. 1 архитектуры для параллельной обработки данных. Далее приводятся только аппаратные конфигурации, обладающие поддержкой набора команд x86 или AMD64 [20], как наиболее распространенные в настоящее время (например, по состоянию на июнь 2013 г. суммарная производительность 500 самых мощных суперкомпьютеров в мире со-

***Таблица 1.*** *Обзор программных систем МГУА, поддерживающих параллельные вычисления*

| Система моделирования | Описание способа распараллеливания | Эксперимент | | | | Поддержка параллельной архитектуры | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Параметры эксперимента | Вычислительная машина | Прирост производительности $\tau_p$ | Эффективность $\eta_p$ | ПП | МЯП | МПС | КЛ | РС |
| FAKE GAME [8, 17] | Распределение нейронов одного слоя сети по вычислительным потокам для параллельного обучения | Выборка данных: 8 переменных, 300 точек. 5 базисов моделей | (2): Intel Core 2 Duo E6550, 2,33 ГГц (8): 2x Intel Xeon E5430, 4 ядра, 2,66 ГГц Java, Threads | (2): 1,65–1,78 (8): 3,36–3,62 | (2): 0,82–8,89 (8): 0,42–0,45 | – | + | + | – | – |
| Knowledge Miner [6, 18] | Распределение моделей, перебираемых в каждом нейроне одного слоя, по вычислительным потокам | Выборка данных: 7 переменных, 10–100000 точек. Дважды многорядная сеть МГУА, двухвходовые нейроны. Полиномиальный базис | 2x Intel Xeon E5472, 4 ядра, 3,0 ГГц, 8 ГБ RAM Apple's Accelerate + Intel Building Blocks | (8) 32 bit: 7,8 64 bit: 7,7 при SIMD оптимиз.: 32 bit: 7,6 64 bit: 7,4 | (8) 32 bit: 0,98 64 bit: 0,96 при SIMD оптимиз: 32 bit: 0,94 64 bit: 0,92 | – | + | + | – | – |
| Parallel Combi [14, 19] | Распределение групп моделей по узлам кластера: (а) «последовательное», (б) «двоичное дополнение» [14] | Комбинаторный алгоритм Полиномиальный базис | HPC Cluster, 26 узлов, Intel Xeon 2,3 ГГц MPI | (26) (а) 14,9 (б) 24,9 | (26) (а) 0,57 (б) 0,96 | – | + | + | + | – |

*ПП – Потоковые процессоры, МЯП – Многоядерные процессоры, МПС – Многопроцессорные системы, КЛ – Кластеры, РС – Распределенные системы, HPC – High Performance Cluster (Высокопроизводительный кластер), MPI – Message Passing Interface (Интерфейс передачи сообщений).*

ставляет 226,7 PFLOPS, из них на долю x86 и AMD64 систем приходится 159,4 PFLOPS, или 71,2 % [21]).

С целью анализа аппаратных конфигураций особенности их архитектуры, включая организацию памяти и оценку производительности, сведены автором в единую табл. 2. Причем информация в данной таблице была структурирована с точки зрения как наглядности восприятия информации, так и полноты анализа. В таблице приводится следующая информация.

1. Примеры систем параллельной обработки данных, соответствующие конкретной реализации каждой архитектуры. При этом рассматриваемые архитектуры расположены в таблице в порядке повышения уровня иерархии. Например, РС могут содержать в своем составе как КЛ, так и МПС. В свою очередь, вычислительные узлы КЛ и МПС основаны на использовании в своем составе МЯП и ПП.
2. Принадлежность к одному из классов параллельных систем, предложенных Э. Таненбаумом [22. С. 38–44]: мультипроцессорам, характеризующимся наличием общей разделяемой памяти, или мультикомпьютерам, имеющим лишь доступную извне локальную память в каждом узле.
3. Принадлежность к одному из классов параллельных архитектур, выделяемых Р. Дунканом в работе [23]: централизованным синхронным системам (SIMD-производные), системам с однородным доступом в разделяемую общую (MIMD-Shared Memory) или частную память узлов (MIMD-Distributed Memory), параллельным гибридным системам (MIMD paradigm).
4. Краткая характеристика вычислительных узлов, составляющих параллельную систему, включая оценку производительности вычислителя (число миллионов операций с плавающей точкой одинарной и двойной точности в секунду), оценку объема локальной памяти и классификацию по М. Флинну [24] (SISD/SIMD/MISD/MIMD – Single/Multiple Instruction stream, Single/Multiple Data stream). Также приводится максимальная суммарная производительность вычислительных узлов в составе параллельной системы (произведение производительности одного узла на максимальное их количество).
5. Оценка времени доступа (латентность – время с момента генерации запроса на данные из памяти до момента их фактического получения), пропускной способности (для удобства приведена в Гбайт/с) и объема памяти параллельной системы.

На основании анализа табл. 2 можно сделать следующие выводы.

1. Наибольшей суммарной производительностью обладают параллельные системы, относящиеся к классу Мультикомпьютеров по классификации Таненбаума [22]. Причиной этому является большее относительно других систем количество вычислительных узлов в составе системы. Недостатком подобных систем является высокая латентность доступа к разделяемой памяти системы, что делает неэффективным распараллеливание относительно небольших вычислительных задач (например, при числе моделей, перебираемых алгоритмом МГУА, сопоставимом с числом вычислительных узлов в системе), поскольку накладные расходы на передачу данных и синхронизацию будут компенсировать прирост производительности от увеличения количества используемых вычислительных мощностей (узлов).
2. Мультипроцессорные системы (особенно ПП и МЯП) позволяют максимизировать эффективность распараллеливания путем минимизации издержек на передачу данных и синхронизацию (по сравнению с КЛ и РС), однако обладают ограниченной производительностью.
3. Все вычислительные узлы классифицированы как SIMD или MIMD [24]. Это дает возможность организовывать «внутреннее распараллеливание» вычислений в каждом узле (low-level parallelism в терминологии Дункана [23]) – применительно к реализации алгоритмов МГУА за счет этого возможен дополнительный прирост производительности путем повышения скорости параметрической оптимизации.
4. Все системы обладают существенно различающейся латентностью локальной памяти вычислительного узла и разделяемой памяти системы (на порядок и более).

На основании данных выводов сформулируем **требования к реализации механизмов параллельных вычислений в рамках платформы МГУА**.

1. Унифицированная поддержка ***всех*** известных параллельных архитектур (ПП, МЯП, МПС, КЛ, РС). С одной стороны, это необходимо для максимального охвата экспериментов по индуктивному моделированию с точки зрения их ресурсоемкости. С другой стороны, это ослабляет ограничения на состав аппаратных вычислительных средств пользователя, требующихся для проведения индуктивного моделирования.
2. Эффективность планирования. Алгоритмы планирования и синхронизации [31. С. 181–190] должны распределять вычислительные задачи в рамках используемой программной системы таким образом, чтобы минимизировать общее время вычислений (в том числе при использовании разнородных систем, например КЛ на основе ПП и МЯП).
3. Эффективность управления памятью. Алгоритмы управления памятью [31. С. 216–230] должны обеспечивать возможность наиболее полного использования локальной памяти вычислительных узлов с целью компенсации высокой латентности доступа в разделяемую память системы.

***Таблица 2.*** *Архитектуры параллельной обработки данных*

| Наименование архитектуры | Архитектура | | | | Вычислительный модуль | | | Суммарная производительность, GFLOPS | Память | | | Примеры реализации |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Вид | Классификация по Танен-бауму [22] | Классификация по Дунка-ну [23] | Число модулей | Производительность модуля, GFLOPS | Классификация по Флинну [24] | Объем локальной памяти на модуль | | Латентность | Пропускная способность, Гбайт/с | Объем, ГБайт | |
| AMD Tahiti [25] | ПП | МП | MIMD-Wavefront | 28–32 ALU: 1792–2048 | [1 ГГц] SP: 128,0 DP: 32,0 ALU: SP: 2,0 DP: 0,5 | SIMD ALU: SISD | Регистры: 64 кБ Разделяемая: 64 кБ L1: 16 кБ | SP: 4096 DP: 1024 | 10 нс | 240–576 | 3 | FirePro W8000, Radeon HD7970 |
| NVIDIA Kepler GK104 [26] | | | | 6–8 ALU: 1152–1536 | [1,05 ГГц] SP: 380,0 DP: 130,0 ALU: SP: 2,0 DP: 0,6 | SIMD ALU: SISD | Регистры: 256 кБ Разделяемая+ L1: 64 кБ | SP: 3090 DP: 1040 | 10 нс | 192–224 | 4 | Quadro K5000, GeForce GTX770, GeForce GTX680 |
| Intel Xeon Phi [27] | | | MIMD-Shared memory | 61 | [1,2 ГГц] SP: 19,8 DP: 9,9 | SIMD | L1: 32+32 кБ L2:512 кБ | SP: 1200 DP: 600 | 10 нс | 352 | 16 | Xeon Phi 7120 |
| AMD K10 [28] | МЯП | | | 2–6 (12 в СМ) | [3 ГГц] (SSE2) SP: 24,0 DP: 12,0 | SIMD | L1: 64+64 кБ L2:512 кБ L3:6144 кБ | SP: <144 DP: <72 | 10 нс | 10,6 | 16 (256 в СМ) | Phenom II 960T |
| AMD Bul-ldozer [15] | | | | 1–4 (16 в СМ) | [4 ГГц] (AVX) SP: 32,0 DP: 16,0 | SIMD/ MIMD | L1: 32+128 кБ L2:2048 кБ L3:8192 кБ | SP: <128 DP: <64 | 10 нс | 14,9 | 32 (256 в СМ) | FX-8170, Opteron 6276 |
| Intel Sandy Bridge [16. С. 36–53] | | | | 1–6 (8 в СМ) | [3,5 ГГц] (SSE2) SP: 28,0 DP: 14,0 | SIMD/ MIMD | L1: 32+32 кБ L2:256 кБ L3:8192 кБ | SP: <168 DP: <84 | 10 нс | 10,6 | 32 (256 в СМ) | i7–2600K , Xeon E5–1600 |
| Intel Haswell [16. С. 31–36] | | | | 2–4 (16 в СМ) | [3,5 ГГц] (SSE2) SP: 28,0 DP: 14,0 | SIMD/ MIMD | L1: 32+32 кБ L2:256 кБ L3:8192 кБ | SP: <112 DP: <56 | 10 нс | 14,9 | 32 | i7–4770 |
| Super Micro 5U | МПС | | MIMD-Distributed memory | 1–8 | SP: 160 | MIMD | <=256 GB | SP: <1280 | 100–200 нс | 10 | 2ТБ | 5086B-TRF |
| Т-Платформы | КЛ | МК | | 640 | DP: 96 | MIMD | 8–48 ГБ | $62 \cdot 10^3$ | 2,5 мкс | 5 | 12 ТБ | СКИФ «Cyberia» |
| Т-Платформы V-Class [29] | | | | 5–10 | DP: 300–320 | MIMD | 256 ГБ | $3{,}2 \cdot 10^3$ | 1,07 мкс | 1,25–5 | 2,56 ТБ | Суперкомпьютер РУДН |
| Т-Платформы T-Blade 2 [30] | | | | 5130 | DP: 140 | MIMD | 24 ГБ | $5 \cdot 10^5$ | 1,07 мкс | 5 | 72,18 ТБ | Суперкомпьютер «Ломоносов» |
| BOINC | РС | | | 1–2 млн | DP: 50–200 | MIMD | в среднем 4 ГБ | $>1 \cdot 10^7$ | 5–400 мс | $10^{-3}$ – 1 | – | Distributed Rainbow Table Generator; PrimeGrid |

*SP – Single Precision (одинарная точность, 32 бит), DP – Double Precision (двойная точность, 64 бит), МК – мультикомпьютеры, МП – мультипроцессоры, СМ – серверная модификация.*

**Реализация параллельных вычислений в рамках программной платформы МГУА**

Диаграмма классов подсистем планирования (TASK) и управления памятью (MEM), полученная в результате объектно-ориентированного анализа и проектирования [32], реализующая основу механизмов параллельных вычислений в рамках платформы МГУА, представлена на рис. 1.

cMemObject – это абстракция, отражающая понятие объекта, выделение и освобождение памяти для которого осуществляется менеджером памяти программной платформы (new (), delete ()). cMemPool – абстракция, объединяющая непосредственно хранилище свободной памяти и объектов cMemObject (_blocks), а также алгоритмы управления ими (менеджер памяти: allocate (), deallocate ()). Стоит отметить, что выделение памяти в данной реализации осуществляется относительно большими блоками cBlock (например, страницы памяти в МЯП и МПС – оптимизация работы на уровне операционной системы). Также предполагается группировка хранимых объектов по типу (например, расположение информации о структурах полиномиальных моделей последовательно в памяти в соответствии с порядком доступа из алгоритма), что потенциально увеличивает эффективность автоматических (МЯП, МПС) и ручных (ПП, КЛ, РС) механизмов управления кэшированием данных в локальной памяти вычислительного узла (пункт 3 требований).

cTask – абстракция вычислительной задачи. Объект класса конкретной вычислительной задачи (производный от интерфейса cTask), будучи созданным, содержит все необходимые данные для осуществления требуемых вычислений, а также реализует алгоритм этих вычислений (например, задача обучения набора полиномиальных моделей на определенной части выборки данных). Реализация менеджера задач cTaskManager ответственна за поддержку иерархии вычислительных задач (каждая задача может делегировать часть вычислений дочерним задачам путем вызова) и за планирование и распределение вычислений по узлам параллельной системы (соответствует пункту 2 требований). При этом каждая вычислительная задача уведомляется о количестве выделенных ей узлов (nThreads) и на каждом узле запускается с уникальным идентификатором (exec (nThreads, id), где id=[0..nThreads-1]).

cThreadManager – вариант реализации менеджера задач для МЯП и МПС, предполагающий динамическое распределение иерархии (дерева) задач по активным потокам (_waiting, _running), привязанным к аппаратным вычислительным узлам. cThread – абстракция вычислительного потока, способная принимать (SetTask ()) на выполнение (Run ()) задачу, назначаемую ей менеджером cThreadManager. cThreadStorage реализует централизованное хранилище потоков (_threads) для согласованной работы нескольких менеджеров задач.

В табл. 3 приведены особенности реализации вычислительных задач cTask для каждого вида па-

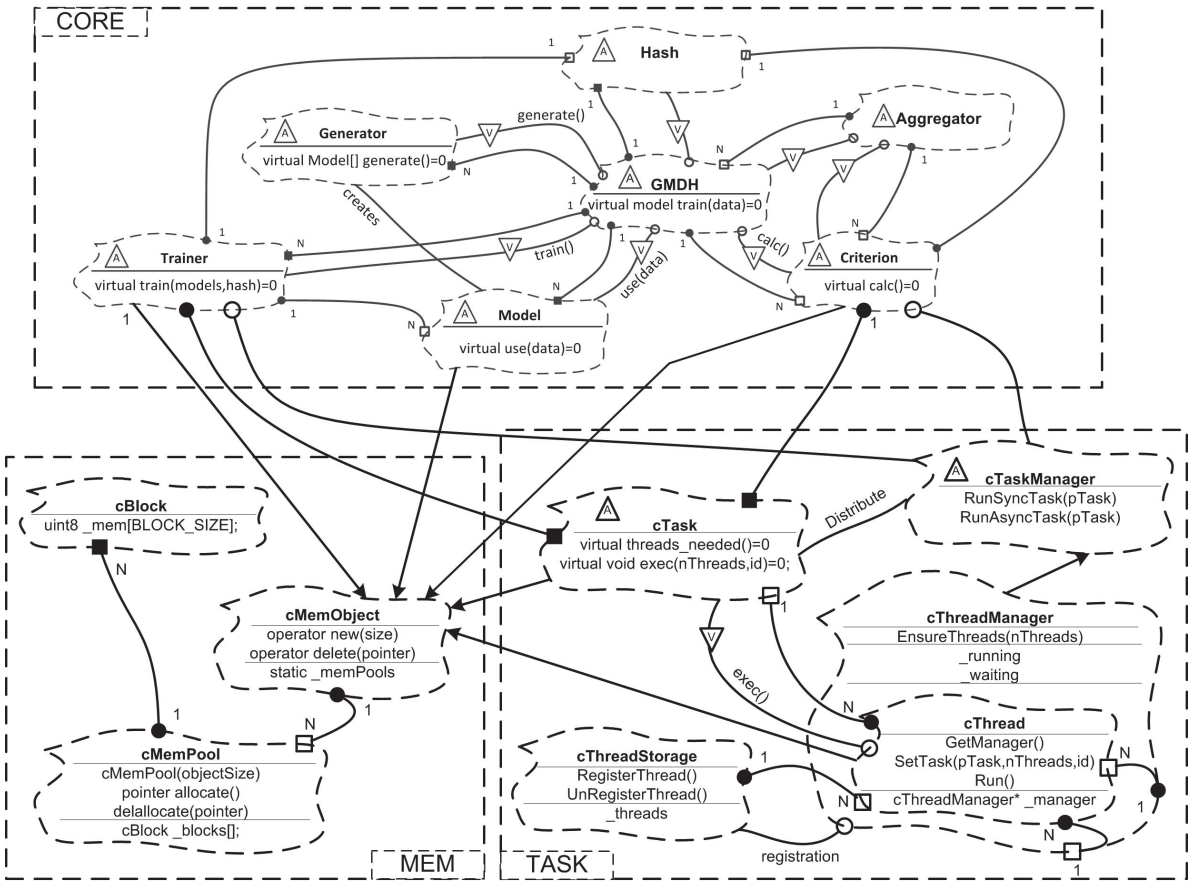


***Рис. 1.*** *Диаграмма классов подсистем управления задачами и памятью*

раллельной архитектуры (в соответствии с пунктом 1 требований) – модель многомерной структуры данных для ПП, многопоточная модель для МЯП и МПС, клиент-серверная модель для КЛ и РС. Также для каждой реализации указаны базовая программная технология и способ передачи данных из разделяемой памяти системы в локальную память вычислительных узлов. Для поддержки запуска вычислительных задач с ожиданием завершения вычислений (для ПП, МЯП, МПС) в интерфейсе класса cTaskManager предусмотрен метод RunSyncTask (), а для поддержки запуска набора независимых вычислительных задач (серверная часть для КЛ и РС при обслуживании нескольких клиентов) представлен метод RunAsyncTask ().

Разработанная в работе [12] совокупность классов, отражающая взаимосвязи сущностей предметной области МГУА и позволяющая реализовывать известные алгоритмы, базисы, методы обучения и селекции моделей, представлена в виде подсистемы CORE и выделена пунктиром.

Организация процесса индуктивного моделирования осуществляется наследниками класса GMDH, реализующими алгоритмы метода. Объекты класса GMDH принимают на вход выборку данных (функция train с параметром data) и производят их обработку (с помощью Hash). Затем они производят генерацию (используя класс Generator), обучение (используя класс Trainer) и селекцию моделей класса Model заданного базиса согласно внешнему критерию или набору критериев (класс Criterion). Класс Aggregator реализует составные алгоритмы МГУА, принимая список используемых (вложенных) алгоритмов МГУА в качестве параметра конструктора. Более подробное описание семантики абстракций подсистемы CORE и связей между ними дано автором в работе [12] и техническом описании [33].

Абстракции Trainer (метод train () – обучение группы моделей Model) и Criterion (метод calc () – расчет значений внешнего критерия) осуществляют распараллеливание вычислений путем создания реализаций абстрактного интерфейса задачи cTask и непосредственного вызова планировщика cTaskManager для организации процесса параллельных вычислений.

### Экспериментальная проверка эффективности предложенной организации параллельных вычислений

Для проверки эффективности предлагаемого подхода к организации параллельных вычислений была проведена реализация подсистем управления задачами и памятью для МЯП (в соответствии с диаграммной классов, изображенной на рис. 1, и особенностей реализации, приведенных в табл. 3). Эксперименты проводились путем включения описанных выше подсистем планирования и управления памятью в состав программной платформы МГУА, используемой в качестве основы системы прогнозирования нестационарных временных рядов с использованием мета-обучения [35]. В качестве исходной информации для проведения индуктивного моделирования использовались данные предыстории нестационарного временного ряда обменного курса валют EUR/USD за 2009–2010 гг. – входная выборка данных размерности 3 (3 входных аргумента), состоящая из 100 точек. Для по-

***Таблица 3.*** *Особенности реализации задач для различных архитектур параллельной обработки данных*

| Вид архитектуры | Клиент | Сервер | Технология | Число потоков | Способ передачи данных |
|---|---|---|---|---|---|
| ПП | Реализация cTask для потоковых процессоров является однопоточной, так как предполагает централизованное формирование пакета данных, передаваемого в память потокового процессора для осуществления параллельных расчетов (выборка данных, описание структур и параметров моделей для обучения и/или селекции). При этом реализация ответственна за определение типа и серии потокового процессора для выбора подходящего ядра (kernel – (в терминологии OpenCL) код, производящий вычисления на стороне потокового процессора [34]). Так же реализация ожидает завершения вычислений и получает результаты из памяти потокового процессора, преобразуя в формат представления основной программы | | OpenCL [34] | 1 | Память потокового процессора |
| МЯП | Многопоточная реализация алгоритмов обучения и селекции совокупности моделей производится путем разделения всего множества моделей на последовательные непересекающиеся части по числу потоков, участвующих в вычислении. При этом процедура разделения может выполняться итеративно с целью балансировки загруженности потоков вычислениями | | Стандартные инструменты синхронизации и работы с потоками многозадачных операционных систем | N | Память процесса |
| МПС | Реализация эквивалентна многоядерным процессорам за исключением необходимости кэширования данных в собственной памяти каждого из процессоров | | | N | Память процесса (копирование) |
| КЛ | Разделение множества моделей на непересекающиеся части для расчета на одном или нескольких серверах. Формирование пакета данных для передачи на сервер, ожидание результатов | Ожидание запроса клиента, запуск задачи обучения или селекции моделей (параллельной, если поддерживается), передача результатов | MPI | N | Сеть передачи данных |
| РС | | | TCP/IP | N | Локальная сеть/ сеть Интернет |

строения прогнозирующих моделей использовался комбинаторный алгоритм МГУА [13. С. 32–38], осуществляющий полный перебор полиномиальных моделей (полиномов) максимальной степени 3 (общее количество моделей составляло 1048575). Параметрическая идентификация моделей проводилась с использованием метода Гаусса (асимптотическая сложность $O(n^3)$). В качестве внешнего критерия селекции моделей использовались критерий точности CR [1. С. 44] и критерий минимума смещения BS [1. С. 54]. Для обучения и селекции моделей выборка делилась на равные части по 50 точек. Для расчета значения CR каждая модель обучалась однократно на одной части, после чего с использованием данных второй части производился отбор 10 лучших моделей, каждая из которых обучалась дважды (на обеих частях выборки) для расчета BS. Эксперимент повторялся несколько раз для различного количества параллельных потоков вычислений (от 1 до 8). Измерялось время, затрачиваемое на вычисления, причем каждый эксперимент повторялся 10 раз с целью минимизации влияния на результаты случайных факторов, увеличивающих время расчета. Для проведения экспериментов использовались три различных аппаратных конфигурации МЯП. Вычисления на каждой из аппаратных конфигураций проводились с использованием вещественных чисел двойной точности (тип double), применялось выравнивание данных по границе, кратной размеру операнда (в соответствии с рекомендациями по оптимизации [15, 16]), векторизация вычислений осуществлялась с применением набора команд SSE2.

На рис. 2 представлены результаты проведенных экспериментов. По оси абсцисс – количество параллельных потоков вычислений $P$, по оси ординат – время вычисления $t_P$. Сплошными линиями и пометкой «64-» обозначены эксперименты, проводимые с использованием 64-битной версии исполняемых файлов, пунктирными линиями и меткой «32-» – 32-битной. Аппаратные конфигурации МЯП, соответствующие меткам «A6», «p2» и «i7», приведены в табл. 4. Стоит отметить, что 32-битная версия исполняемого файла показала меньшую производительность, чем 64-х битная. Это может быть объяснено наличием большего количества регистров общего назначения, доступных приложениям в программной модели AMD64 по сравнению с x86 [20. С. 1–8].

На рис. 3, *а* приведены графики прироста производительности $\tau_P$ (ось ординат) в зависимости от числа потоков вычисления $P$ (ось абсцисс) для различных аппаратных конфигураций. Наибольший прирост наблюдается для 6-ядерного процессора – в 3,96 раза. Что характерно, линейный характер увеличения производительности с ростом числа потоков исчезает при достижении количества аппаратно-доступных ядер.

На рис. 3, *б* приведены графики эффективности распараллеливания $\eta_P$. Видно, что при количестве потоков вычисления, совпадающем с количеством аппаратных ядер, эффективность составляет 0,63–0,67 для всех конфигураций.

***Таблица 4.*** *Аппаратные конфигурации МЯП, использованные при проведении экспериментов*

| Метка графиков на рис. 2 | Процессор | Память | Операционная система |
|---|---|---|---|
| p2 | AMD Phenom II 960T, K10, 6 cores (4+2 unlocked), 3 ГГц | 8GB, DDR3–1333, 9-9-9-24, 13 нс | Windows 7, 64 bit |
| i7 | Intel Core i7–2600K, Sandy Bridge, 4 cores, HT, 3,4 ГГц | 4GB, DDR3–1333, 9-9-9-24, 13 нс | Windows 7, 32 bit |
| A6 | AMD A6–6400K, Richland/Piledriver, 2 cores, HD8470D/800МГц, 3,9 ГГц | 4GB, DDR3–1600, 9-9-9-27, 15 нс | Windows 7, 64 bit |

На рис. 4 приведены графики, отображающие время, затраченное каждым потоком вычисления на расчет определенной ему части общей вычислительной задачи. Каждый из графиков соответствует определенному числу потоков – от 2 до 8, причем подписи под осью абсцисс определяют индекс id вычислительного потока (условный номер вычислительного потока в диапазоне от 1 до числа максимального количества потоков). Высота столбцов пропорциональна затраченному времени (каждый из столбцов группы соответствует своей аппаратной конфигурации и битности исполняемого файла).

Видно, что распределение вычислительной нагрузки производится неравномерно – некоторые потоки (с меньшими индексами) остаются «недозагруженными» вычислениями, поэтому определенную часть времени простаивают в ожидании завершения вычислений на других потоках. Причиной этому являются неточности в предсказании времени обучения моделей в текущей реализации алгоритма распределения моделей по независимым группам. Увеличение точности предсказания является задачей дальнейших работ. Тем не менее, на основании текущих экспериментальных данных можно произвести оценку максимальной производительности предложенного решения. Минимальное общее время расчета соответствует равномерному распределению вычислительной нагрузки. В этом случае каждый поток затратит время, равное среднему измеренному времени работы потоков. Графики прироста производительности и эффективности в этом случае для МЯП «p2» приведены на рис. 3 и помечены «ideal» – для 6 потоков прирост составит 5,77, а эффективность – 0,96, что соответствует лучшим показателям программных систем индуктивного моделирования, рассмотренным выше (табл. 1).

#### Заключение

Гибкость программной платформы МГУА, предложенной автором в работе [12], позволила без какого-либо изменения ее объектно-ориентиро-

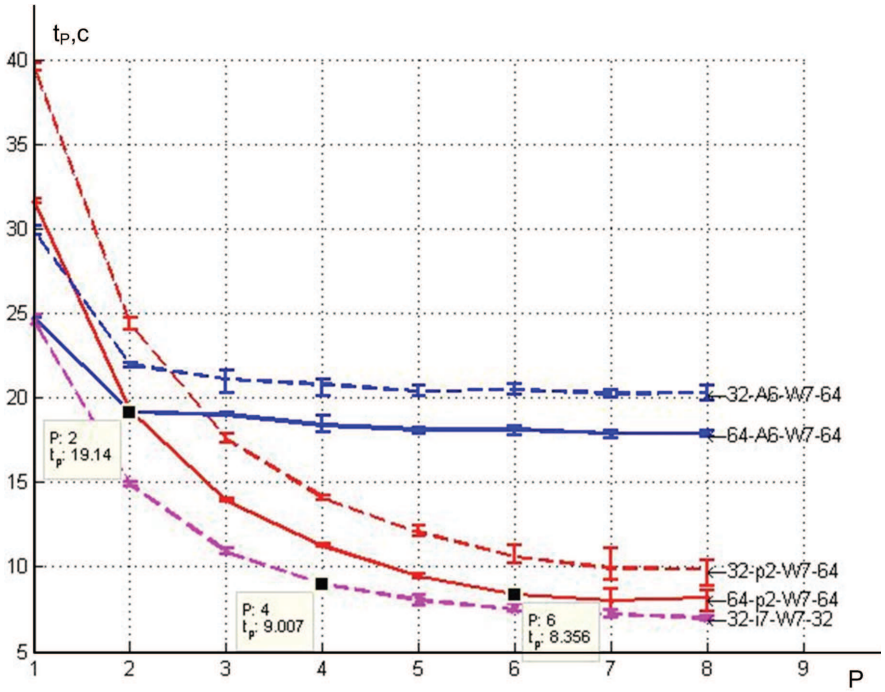


***Рис. 2.*** *Графики времени выполнения расчета $t_p$ при различном количестве потоков вычислений P для различных аппаратных конфигураций МЯП*

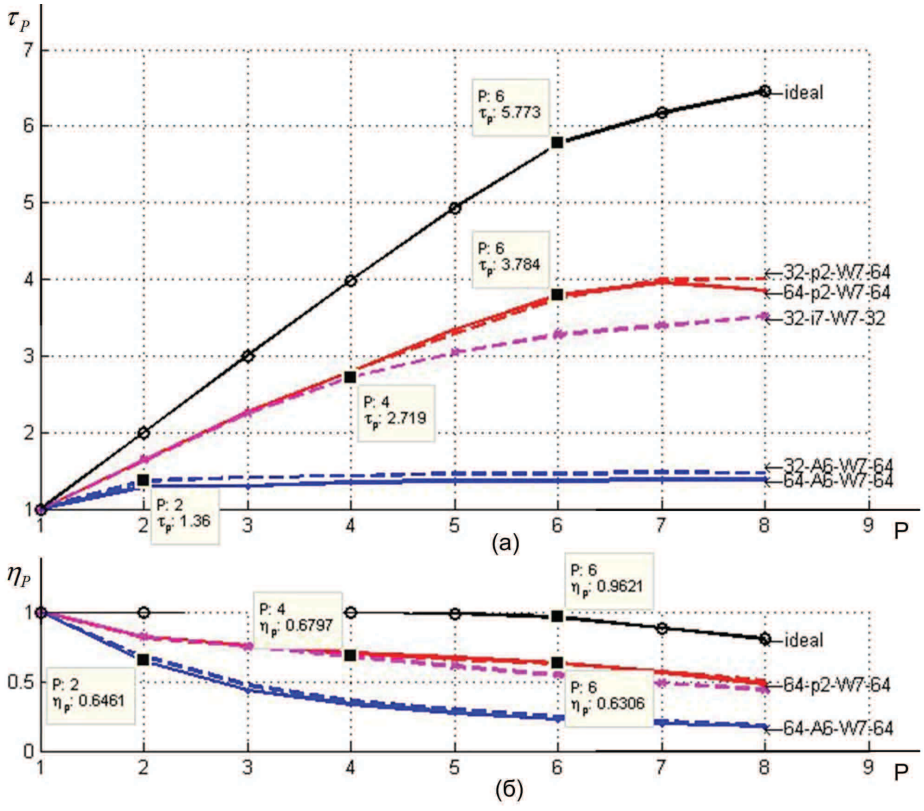


***Рис. 3.*** *Графики прироста производительности $\tau_P$ (а) и эффективности распараллеливания $\eta_P$ (б) при различном количестве потоков вычислений P для различных аппаратных конфигураций МЯП*

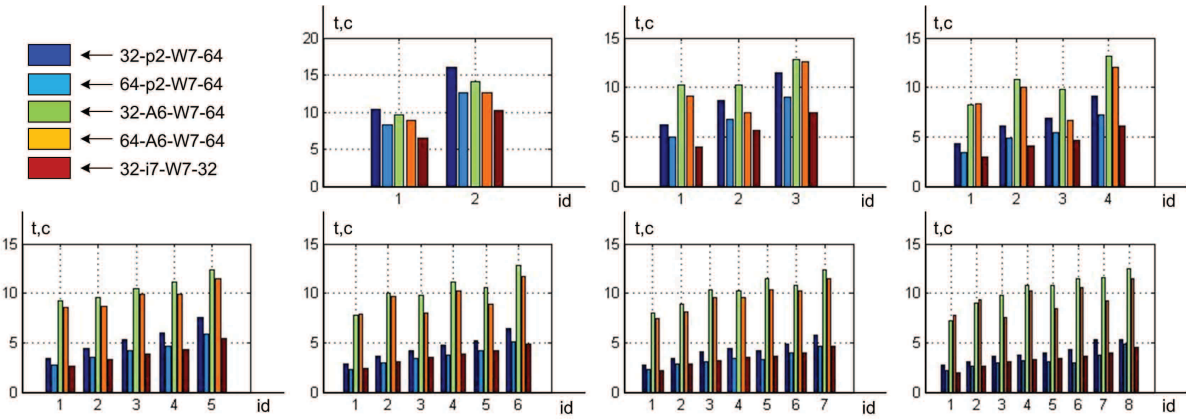


***Рис. 4.*** *Распределение времени выполнения расчетов по вычислительным потокам при различном их количестве*

ванной структуры (архитектуры) реализовать в ее составе подсистемы параллельных вычислений и управления памятью, позволяющих увеличить производительность любого алгоритма МГУА, реализуемого на базе программной платформы. Экспериментально полученные значения прироста производительности и коэффициента распараллеливания для комбинаторного алгоритма МГУА для различных аппаратных конфигураций МЯП позволяют сделать вывод о том, что предложенная программная платформа МГУА удовлетворяет сформулированному требованию производительности. Направлением дальнейших работ является увеличение точности предсказания времени обучения моделей в реализации для МЯП и МПС, а также перенос части вычислений на ПП.

*Работа выполнена в рамках Гос. задания Министерства Образования и Науки РФ на 2013 г. (проект № 7.2868.2011).*

*Автор выражает благодарность Анатолию Андраханову, сотруднику лаборатории Интеллектуальных Систем, Когнитивной Робототехники и Автоматизации (ИСКРА) кафедры Промышленной Электроники Томского Государственного Университета Систем Управления и Радиоэлектроники, за ценные замечания в процессе подготовки рукописи статьи.*

## СПИСОК ЛИТЕРАТУРЫ

1. Ивахненко А.Г. Индуктивный метод самоорганизации моделей сложных систем. – Киев: Наукова Думка, 1982. – 296 с.
2. Anastasakis L., Mort N. The development of self-organization techniques in modelling: A review of the Group Method of Data Handling (GMDH): Research Report № 813. – Sheffield, United Kingdom: The University of Sheffield, 2001. – 38 p.
3. Ivakhnenko A.G., Ivakhnenko G.A. The Review of Problems Solvable by Algorithms of the Group Method of Data Handling // International Journal of Pattern Recognition and Image Analysis: Advanced in Mathematical Theory and Applications. – 1995. – V. 5. – № 4. – P. 527–535.
4. GMDH – Examples of Applications // Group Method of Data Handling (GMDH). 2010. URL: http://www.gmdh.net/GMDH_exa.htm (дата обращения: 05.09.2013).
5. Степашко В.С., Булгакова А.С. Обобщенный итерационный алгоритм метода группового учета аргументов // Управляющие системы и машины: Международный журнал. – 2013. – № 2. – С. 5–17.
6. Muller J.A., Lemke F. Self-Organising Data Mining. An Intelligent Approach to Extract Knowledge from Data. – Dresden, Berlin: Books on Demand GmbH, 1999. – 225 p.
7. Ivakhnenko A.G., Kovalishyn V.V., Tetko I.V., Luik A.I. et al. Self-Organization of Neural Networks with Active Neurons for Bioactivity of Chemical Compounds Forecasting by Analogues Complexing GMDH algorithm: Poster for the 9th International Conference on Neural Networks (ICANN 99). – Edinburg, 1999. – 13 p. URL: http://www.gmdh.net/ articles/ papers/ nn_anal.pdf (дата обращения: 05.09.2013).
8. Kordik P. Fully Automated Knowledge Extraction using Group of Adaptive Models Evolution: PhD thesis. – Prague, 2006. – 136 p.
9. GMDH Shell – Forecasting Software for Professionals. Прикладная программа. URL: http://www.gmdhshell.com (дата обращения: 05.09.2013).
10. Степашко В.С. Основные требования к функциональной структуре ППП МГУА для персональных ЭВМ // Управление в технических системах. – Киев: ИК АН УССР, 1990. – С. 27–34.
11. Genetic Design of GMDH-type Neural Networks for Modelling of Thermodynamically Pareto Optimized Turbojet Engines / K. Atashkari, N. Nariman-zadeh, A. Darvizeh, X. Yao, A. Jamali, A. Pilechi // WSEAS Transactions on COMPUTERS. – 2004. – V. 3. – Iss. 3. – P. 177–183.
12. Орлов А.А. Принципы построения архитектуры программной платформы для реализации алгоритмов метода группового учета аргументов (МГУА) // Управляющие системы и машины: Международный журнал. – 2013. – № 2. – С. 65–71.
13. Madala H.R., Ivakhnenko A.G. Inductive Learning Algorithms for Complex System Modeling. – Boca Raton; Ann Arbor; London; Tokyo: CRC Press, 1994. – 368 p.
14. Koshulko O.A., Koshulko A.I. Adaptive parallel implementation of the Combinatorial GMDH algorithm // Proceedings of International Workshop on Inductive Modelling (IWIM-2007). – Prague, 2007. – P. 71–77.
15. Software Optimization Guide for AMD Family 15h Processors. V. 3.06. Advanced Micro Devices (AMD). – 2012. – 362 c. URL: http://support.amd.com/ us/ Processor_TechDocs/ 47414_15h_sw_opt_guide.pdf (дата обращения: 05.09.2013).
16. Intel® 64 and IA-32 Architectures. Optimization Reference Manual: справочное руководство. V. 3.06. Intel Corporation. – 2013. – 670 с. URL: http://www.intel.com/ content/ dam/ www/ public/ us/ en/ documents/ manuals/ 64-ia-32-architectures-optimization-manual.pdf (дата обращения: 05.09.2013).

17. Kordik P., Spirk J., Simecek I. Parallel computing of GAME models // Proceedings of 2nd International Conference on Inductive Modelling (ICIM-2008). – Kyiv, 2008. – P. 160–163.
18. Lemke F. Parallel Self-organizing Modeling // Proceedings of 2nd International Conference on Inductive Modelling (ICIM-2008). – Kyiv, 2008. – P. 176–183.
19. Koshulko O.A., Koshulko A.I. Acceleration of GMDH combinatorial search with HPC clusters // Proceedings of 2nd International Conference on Inductive Modelling (ICIM-2008). – Kyiv, 2008. – P. 164–167.
20. AMD64 Technology. AMD64 Architecture. В 5 ч. Ч. 1. Application Programming. V. 3.20. Advanced Micro Devices (AMD). – 2013.– 386 с. URL: http://support.amd.com/ us/ Processor_TechDocs/ 24592_APM_v1.pdf (дата обращения: 05.09.2013).
21. TOP500. List Statistics. Process generation category. June 2013 (статистика по поколениям процессоров, использованных в суперкомпьютерах списка Top 500). – TOP500 Supercomputers. 2013. URL: http://www.top500.org/statistics/list/ (дата обращения: 05.09.2013).
22. Таненбаум Э., М. ван Стеен. Распределенные системы. Принципы и парадигмы. – СПб.: Питер, 2003. – 887 с.
23. Duncan R. A survey of parallel computer architectures // Computer: Journal of IEEE. – 1990. – V. 23. – Iss. 2. – P. 5–16.
24. Flynn M.J. Very High-Speed Computing Systems // Proceedings of the IEEE. – 1966. – V. 54. – P. 1901–1909.
25. AMD Graphics Cores Next (GCN) architecture: техническое описание. V. 1.0. Advanced Micro Devices Inc. – 2012. – 18 с. URL: http://www.amd.com/ us/ Documents/ GCN_Architecture_whitepaper.pdf (дата обращения: 05.09.2013).
26. NVIDIA GeForce GTX 680: техническое описание. V. 1.0. NVIDIA. – 2012. – 29 с. URL: http://international.download.nvidia.com/ webassets/ en_US/ pdf/ GeForce-GTX-680-Whitepaper-FINAL.pdf (дата обращения: 05.09.2013).
27. Chrysos G. Intel Xeon Phi Coprocessor – the Architecture: техническое описание. Intel Corporation. 2012. URL: http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner (дата обращения: 05.09.2013).
28. Software Optimization Guide for AMD Family 10h and 12h Processors. V. 3.13. Advanced Micro Devices (AMD). – 2011. – 348 с. URL: http://support.amd.com/ us/ Processor_TechDocs/ 40546.pdf (дата обращения: 05.09.2013).
29. Семейство T-Blade V-Class. Обзор вычислительных модулей V205S и V205F. Т-Платформы. – 2012. – 24 с. URL: http://www.t-platforms.ru/ images/ pdfvclass_v5000_RUS/ Обзор вычислительных модулей V205S и V205F.pdf (дата обращения: 05.09.2013).
30. T-Blade 2: техническое руководство. Rev. A02. Т-Платформы. – 2010. – 24 с. URL: http://www.t-platforms.ru/ images/ pdfblade2_products_RUS/ TB2 Техническое руководство.pdf (дата обращения: 05.09.2013).
31. Таненбаум Э. Современные операционные системы. 3-е изд. – СПб.: Питер, 2010. – 1120 с.
32. Object-Oriented Analysis and Design with Applications / G. Booch, R.A. Maksimchuk, M.W. Engel, B.J. Young, J. Conallen, K.A. Houston. 3d ed. – NY: Addison-Wesley Professional, 2007. – 720 p.
33. Орлов А.А. Электронный информационный образовательный ресурс: <Программная платформа для реализации метода группового учета аргументов> // Хроники объединенного фонда электронных ресурсов <Наука и образование>. – 2013. – № 2. URL: http://ofernio.ru/ portal/ newspaper/ ofernio/ 2013/ 2.doc (дата обращения: 05.09.2013).
34. The OpenCL Specification / под ред. A. Munshi. Ver. 1.2., Rev. 19. Khronos OpenCL Working Group. – 2012. – 380 p. URL: http://www.khronos.org/ registry/ cl/ specs/ opencl-1.2.pdf (дата обращения: 05.09.2013).
35. Orlov A.A. Non-Stationary Time Series Forecasting on Basis of Analysis and Prediction of Forecasting Models Efficiency // Proceedings of 4th International Conference on Inductive Modelling (ICIM-2013). – Kyiv, 2013. – P. 192–199.

*Поступила 11.09.2013 г.*

UDC 004.41[2+4+5]::004.272::004.855

# INCREASING THE PERFORMANCE OF THE SOFTWARE FRAMEWORK FOR IMPLEMENTING THE ALGORITHMS OF THE GROUP METHOD OF DATA HANDLING

A.A. Orlov

Tomsk State University of Control Systems and Radioelectronics

*In previous works the author has proposed a universal software framework that allows implementing the known algorithms of group method of data handling, model bases, training methods and model selection criteria. This paper introduces the solution of a topical problem of increasing the performance of the framework. Based on the review of existing computing architectures for parallel data processing and software systems for inductive modeling supporting parallel computations the author has worked out the requirements for the subsystems of parallel computing and memory management of the software framework. Using the methodology of object-oriented analysis and design the author developed the object-oriented structure of these subsystems and introduced the specifics of their operation on each of the mentioned computing architectures. The performance of the parallel implementation of the combinatorial group method of data handling algorithm on basis of the software framework was evaluated experimentally for multi-core processors.*

***Key words:***
*Software framework, object-oriented analysis and design, parallel data processing, efficiency of paralleling, group method of data handling.*

## REFERENCES


1. Ivakhnenko A.G. *Induktivny metod samoorganizatsii modeley slozhnykh system* [Inductive method of self-organization of models of complex systems]. Kiev, Naukova Dumka, 1982. 296 p.
2. Anastasakis L., Mort N. *The development of self-organization techniques in modelling: A review of the Group Method of Data Handling (GMDH): Research Report № 813*. Sheffield, United Kingdom, The University of Sheffield, 2001. 38 p.
3. Ivakhnenko A.G., Ivakhnenko G.A. The Review of Problems Solvable by Algorithms of the Group Method of Data Handling. *International Journal of Pattern Recognition and Image Analysis: Advanced in Mathematical Theory and Applications*, 1995, vol. 5, no. 4, pp. 527–535.
4. *GMDH – Examples of Applications*. Group Method of Data Handling (GMDH). 2010. Available at: http://www.gmdh.net/ GMDH_exa.htm (accessed 05 September 2013).
5. Stepashko V.S., Bulgakova A.S. Obobshhenny iteratsionny algoritm metoda gruppovogo ucheta argumentov [Generalized iterative algorithm of group method of data handling]. *Control Systems and Computers: International Journal*, 2013, no. 2, pp. 5–17.
6. Muller J.A., Lemke F. *Self-Organising Data Mining. An Intelligent Approach to Extract Knowledge from Data*. Dresden, Berlin, Books on Demand GmbH, 1999. 225 p.
7. Ivakhnenko A.G., Kovalishyn V.V., Tetko I.V., Luik A.I. Self-Organization of Neural Networks with Active Neurons for Bioactivity of Chemical Compounds Forecasting by Analogues Complexing GMDH algorithm. *Poster for the 9th International Conference on Neural Networks (ICANN 99)*. Edinburg, 1999. 13 p. Available at: http://www.gmdh.net/ articles/ papers/ nn_anal.pdf (accessed 05 September 2013).
8. Kordik P. *Fully Automated Knowledge Extraction using Group of Adaptive Models Evolution: PhD thesis*. Prague, 2006. 136 p.
9. *GMDH Shell – Forecasting Software for Professionals*. Application program. Available at: http://www.gmdhshell.com (accessed 05 September 2013).
10. Stepashko V.S. Osnovnye trebovaniya k funktsionalnoy strukture PPP MGUA dlya personalnykh JeVM [Basic requirements for functional structure of GMDH software package for Personal Computers]. *Upravlenie v tekhnicheskikh sistemakh*. Kiev, IK AN USSR, 1990. pp. 27–34.
11. Atashkari K., Nariman-zadeh N., Darvizeh A., Yao X., Jamali A., Pilechi A. Genetic Design of GMDH-type Neural Networks for Modelling of Thermodynamically Pareto Optimized Turbojet Engines. *WSEAS Transactions on COMPUTERS*, 2004, vol. 3, Iss. 3, pp. 177–183.
12. Orlov A.A. Printsipy postroeniya arkhitektury programmnoy platformy dlya realizatsii algoritmov metoda gruppovogo ucheta argumentov (MGUA) [Principles of construction of the software framework architecture for implementation of the algorithms of group method of data handling (GMDH)]. *Control Systems and Computers: International Journal*, 2013, no. 2, pp. 65–71.
13. Madala H.R., Ivakhnenko A.G. *Inductive Learning Algorithms for Complex System Modeling*. Boca Raton; Ann Arbor; London; Tokyo: CRC Press, 1994. 368 p.
14. Koshulko O.A., Koshulko A.I. Adaptive parallel implementation of the Combinatorial GMDH algorithm. *Proc. International Workshop on Inductive Modelling (IWIM-2007)*, Prague, 2007, pp. 71–77.
15. *Software Optimization Guide for AMD Family 15h Processors. V. 3.06. Advanced Micro Devices (AMD)*. 2012, 362 p. Available at: http://support.amd.com/ us/ Processor_TechDocs/ 47414_15h_ sw_opt_guide.pdf (accessed 05 September 2013).
16. *Intel® 64 and IA-32 Architectures. Optimization Reference Manual*. Vol. 3.06. Intel Corporation, 2013, 670 p. Available at: http://www.intel.com/ content/ dam/ www/ public/ us/ en/ documents/ manuals/ 64-ia-32-architectures-optimization-manual.pdf (accessed 05 September 2013).
17. Kordik P., Spirk J., Simecek I. Parallel computing of GAME models. *Proc. the 2nd International Conference on Inductive Modelling (ICIM-2008)*. Kyiv, 2008. pp. 160–163.
18. Lemke F. Parallel Self-organizing Modeling. *Proc. the 2nd International Conference on Inductive Modelling (ICIM-2008)*. Kyiv, 2008, pp. 176–183.
19. Koshulko O.A., Koshulko A.I. Acceleration of GMDH combinatorial search with HPC clusters. *Proc. the 2nd International Conference on Inductive Modelling (ICIM-2008)*. Kyiv, 2008, pp. 164–167.
20. *AMD64 Technology. AMD64 Architecture*. Vol. 5 ch. Ch. 1. Application Programming. Vol. 3.20. Advanced Micro Devices (AMD). 2013. 386 p. Available at: http://support.amd.com/ us/ Processor_TechDocs/ 24592_APM_v1.pdf (accessed 05 September 2013).
21. *TOP500. List Statistics. Process generation category*. June 2013: (statistics on processor generations used in supercomputers from the Top500 list). – TOP500 Supercomputers. 2013. Available at: http://www.top500.org/statistics/list/ (accessed 05 September 2013).
22. Tanenbaum E., M. van Steen. *Raspredelennye sistemy. Printsipy i paradigmy* [Distributed Systems: Principles and Paradigms]. Saint Petersburg, Piter, 2003. 887 p.
23. Duncan R. A survey of parallel computer architectures. *Computer: Journal of IEEE*, 1990, vol. 23, Iss. 2, pp. 5–16.
24. Flynn M.J. Very High-Speed Computing Systems. *Proc. the IEEE*, 1966, vol. 54, pp. 1901–1909.
25. *AMD Graphics Cores Next (GCN) architecture. V. 1.0. Advanced Micro Devices Inc*. 2012. 18 p. Available at: http://www.amd.com/ us/ Documents/ GCN_Architecture_whitepaper.pdf (accessed 05 September 2013).
26. *NVIDIA GeForce GTX 680. V.1.0. NVIDIA*. 2012. 29 p. Available at: http://international.download.nvidia.com/ webassets/ en_US/ pdf/ GeForce-GTX-680-Whitepaper-FINAL.pdf (accessed 05 September 2013).
27. Chrysos G. *Intel Xeon Phi Coprocessor – the Architecture*. Intel Corporation. 2012. Available at: http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner (accessed 05 September 2013).
28. *Software Optimization Guide for AMD Family 10h and 12h Processors*. V.3.13. Advanced Micro Devices (AMD). 2011. 348 p. Available at: http://support.amd.com/ us/ Processor_TechDocs/ 40546.pdf (accessed 05 September 2013).
29. *Semeystvo T-Blade V-Class. Obzor vychislitelnykh moduley V205S i V205F. T-Platformy* [Review of computation units V205S and V205F. T-Platforms]. 2012. 24 p. Available at: http://www.t-platforms.ru/ images/ pdfvclass_v5000_RUS/ Obzor vychislitel'nyh module V205S i V205F.pdf (accessed 05 September 2013).
30. *T-Blade 2. Rev.A02. T-Platforms*. 2010. 24 p. Available at: http://www.t-platforms.ru/ images/ pdfblade2_products_RUS/ TB2 Tehnicheskoe rukovodstvo.pdf (accessed 05 September 2013).
31. Tanenbaum E. *Sovremennye operatsionnye sistemy* [Modern Operating Systems]. Saint Petersburg, Piter, 2010. 1120 p.
32. Booch G., Maksimchuk R.A., Engel M.W., Young B.J., Conallen J., Houston K.A. *Object-Oriented Analysis and Design with Applications*. NY, Addison-Wesley Professional, 2007. 720 p.
33. Orlov A.A. Elektronny informatsionny obrazovatelny resurs <Programmnaya platforma dlya realizatsii metoda gruppovogo ucheta argumentov> [Electronic informational educational resource <Software platform for implementation of algorithms of group method of data handling (GMDH)>]. *Hroniki obedinennogo fonda elektronnykh resursov <Nauka i obrazovanie>* [Chronicles of the united fund of electronic resources <Science and Education>], 2013, no. 2. Available at: http://ofernio.ru/ portal/ newspaper/ ofernio/ 2013/ 2.doc (accessed 05 September 2013).
34. Munshi A. *The OpenCL Specification*. Ver.1.2., Rev.19. Khronos OpenCL Working Group. 2012. 380 p. Available at: http://www.khronos.org/ registry/ cl/ specs/ opencl-1.2.pdf (accessed 05 September 2013).
35. Orlov A.A. Non-Stationary Time Series Forecasting on Basis of Analysis and Prediction of Forecasting Models Efficiency. *Proc. the 4th International Conference on Inductive Modelling (ICIM-2013)*. Kyiv, 2013. pp. 192–199.