УДК 004.4'24

А.Ю. Дорошенко, О.Г. Бекетов, К.А. Жереб, П.А. Іваненко, О.М. Овдій, Р.С. Шевченко, О.А. Яценко

ФОРМАЛЬНІ ТА АДАПТИВНІ МЕТОДИ Й ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ

Наведено огляд результатів розробки формальних та адаптивних методів і програмних засобів, досягнутих за останні роки в Інституті програмних систем НАН України, що ґрунтуються на алгебро-алгоритмічному підході та техніці переписувальних правил. Методи призначені для автоматизації проектування, генерації та перетворень паралельних програм для широкого діапазону мультипроцесорних обчислювальних платформ і знаходять застосування для різних прикладних областей, зокрема, для ефективної реалізації задач метеорологічного прогнозування.

Ключові слова: автоматизація програмування, алгебра алгоритмів, метеорологічне прогнозування, паралельні обчислення, техніка переписувальних правил.

Вступ

Паралельні обчислення на мультипроцесорних системах на сьогодні є основним джерелом забезпечення необхідних потреб у високій продуктивності обчислень при розв’язуванні складних науково-технічних і господарських проблем. Подальший прогрес у покращенні показників якості створення паралельних систем пов’язаний не тільки з підвищенням продуктивності їх роботи, а й із зменшенням споживаної енергії на одиницю обчислювальної потужності ("зелені" обчислення). Досягнення таких цілей лежить на шляху до використання архітектур паралельних обчислювальних систем, здатних до спеціалізації обчислень для конкретних класів задач з метою отримання найвищих показників продуктивності при найменших ресурсних витратах. За останні роки у цьому відношенні найкраще себе проявили, наприклад, графічні прискорювачі (graphics processing units, GPU) – спеціалізовані недорогі мультипроцесорні системи, спроектовані спочатку як додаткові до центральних процесорів (CPU) засоби обробки відеоінформації і використані потім як мультипроцесорні платформи загального призначення (GPGPU). Проте, розробка програм для графічних прискорювачів є досить складною задачею, тому постає питання щодо розробки спеціальних засобів автоматизації розробки програмного забезпечення, що дозволяли б найбільш ефективно генерувати найпродуктивніший код для таких систем.

На даний час Грід-системи та "хмарні" обчислення все більше використовуються як в наукових, так і в промислових застосуваннях. Такі системи використовують низку взаємно пов'язаних ідей, включаючи поняття загальнодоступних обчислень, сервісів, Грід-технологій, віртуальних ресурсів та ін., спрямованих на граничне абстрагування ресурсів та спрощення доступу до них з боку користувача. Вони дозволяють ефективно використовувати існуючі різнорідні паралельні ресурси та вирішувати задачі великих обсягів, які не можуть бути вирішені на окремих паралельних комплексах. Однак, проблемою залишається оптимізація затрат і зусиль на отримання ефективного програмного коду для таких систем.

У відділі теорії комп’ютерних обчислень Інституту програмних систем НАН України упродовж тривалого періоду розвивається напрямок досліджень, пов’язаний з розробкою теорії, методології та інструментальних засобів для автоматизованого проектування паралельних і розподілених програм. Даний напрямок бере свій початок від праць В.М. Глушкова, Г.О. Цейтліна та К.Л. Ющенко з розробки систем алгоритмічних алгебр та їх модифі-

кацій [1–3] і праць О.А. Летичевського та його школи з алгебраїчного програмування [4–6], що ґрунтується на техніці переписування термів.

Мета даної роботи – огляд результатів, отриманих в рамках розробки формальних методів та програмних засобів на основі алгеброалгоритмічного підходу та техніки переписувальних правил для автоматизації програмування мультипроцесорних платформ [7–29].

1. Формальні методи й інструментарій проектування та генерації програм для графічних прискорювачів

В основу пропонованого підходу до проектування паралельних програм покладений апарат модифікованих систем алгоритмічних алгебр (САА-М) [2, 7], призначений для формалізації процесів мультиобробки, що виникають при конструюванні програмного забезпечення в мультипроцесорних системах.

1.1. Алгебра алгоритмів. САА-М – двоосновна алгебра $< Op, Pr; \Omega >$, де Op – множина операторів; Pr – множина логічних умов; Ω – сигнатура, яка складається з логічних операцій (диз'юнкції, кон'юнкції, заперечення, лівого множення оператора на умову) та операторних операцій (композиції, альтернативи, циклу та ін.) [7]. Алгоритми в САА-М можуть бути подані в трьох формах: аналітичній (регулярні схеми), природньо-лінгвістичній (САА-схеми) та графовій (граф-схеми). У роботах [11–13] виконаний подальший розвиток САА-М в напрямку проектування паралельних програм для графічних прискорювачів, що використовують технологію Nvidia CUDA [30].

Далі наведено перелік назв та специфікації операторних операцій сигнатури САА-М, що використовуються у даній роботі і подані у природньо-лінгвістичній формі.

1. Композиція (послідовне виконання) операторів:

"*оператор* 1"; "*оператор* 2".

2. Операція виконання одного з n операторів за істинності відповідної умови:

ВИБІР (['*умова* 1'] → "*оператор* 1",
...
['*умова n*'] → "*оператор n*").

3. Асинхронна диз'юнкція – паралельне виконання двох операторів (потоків):

"*оператор* 1"
ПАРАЛЕЛЬНО
"*оператор* 2".

4. Синхронізатор, що виконує затримку обчислень доти, поки значення умови не стане істинним:

ЧЕКАТИ '*умова*'.

5. Операція виклику функції-ядра у схемах алгоритмів для графічних прискорювачів:

Запуск Ядра (*nB*, *nTh*)
("*оператор*"),

де *nB* – кількість блоків потоків; *nTh* – кількість потоків у кожному блоці; "*оператор*" – частина алгоритму, що буде виконуватися паралельно.

Застосування цих операцій продемонстроване у розділах 5 та 7.

1.2. Методи й інструментарій автоматизованої розробки програм. Використання апарату САА-М покладено в основу розробленої методології та інструментальних засобів підтримки проектування та генерації програм. Інструментарій ґрунтується на методі діалогового конструювання синтаксично правильних програм (ДСП-методі) [7], що орієнтований на виключення можливості появи синтаксичних помилок у процесі побудови алгоритму. Основна ідея методу полягає у порівневому конструюванні алгоритмів зверху вниз шляхом суперпозиції мовних конструкцій САА-М. На основі побудованої схеми алгоритму виконується автоматична генерація тексту програми цільовою мовою програмування.

ДСП-метод був реалізований у системі ІПС [7, 8, 14, 15], яка містить такі основні компоненти:

- діалоговий конструктор синтаксично правильних програм (ДСП-конструктор), призначений для діалогового про-

ектування схем алгоритмів та синтезу програм мовами Java, C++, C++ для CUDA;

- редактор граф-схем;
- база даних алгеброалгоритмічних специфікацій, у якій зберігається текст конструкцій САА-М і базисних елементів схем, а також їх програмні реалізації;
- генератор САА-схем за гіперсхемами [15] (див. також розділ 5).

Для автоматизації виконання трансформацій алгоритмів система ІПС застосовується спільно з системою TermWare [8, 9, 14], яка ґрунтується на парадигмі переписувальних правил.

У роботах [12, 13] розглядається нова версія системи ІПС, названа онлайновим діалоговим конструктором синтаксично правильних програм (ОДСП), особливістю якої є сервісно-орієнтована архітектура та спрямованість на багатокористувальницьке використання системи через Інтернет.

2. Засоби опису паралельних алгоритмів у рамках алгебри алгоритмів з даними для класу інформаційно-керуючих систем

З огляду на важливість ролі, яку відіграють інформаційно-керуючі системи (ІКС) у сучасному світі, в роботах [16, 17] виконана розробка засобів специфікації та перетворення паралельних алгоритмів для таких систем. Для опису алгоритмів застосовано алгебраїчний апарат, побудований у результаті модифікації відомої моделі ЕОМ Глушкова. Модифікація полягає у доповненні згаданої моделі зовнішнім середовищем (ЗС), що складається з програмної та апаратної складових. Програмна складова забезпечує взаємодію між моделями, що функціонують у різних ЗС (багатомашинна організація), і між задачами в рамках одного ЗС (квазіпаралельна, багатозадачна організація). Апаратна складова є пам'яттю та зовнішніми пристроями, які включають пристрої введення-виведення та пристрої зв'язку з об'єктом керування, і інтерпретуються як дані, що формалізуються таким чином.

Даними називається пара $D = \langle N, 3 \rangle$, де N – носій даних, 3 – кортеж значень, носієм яких є N. На кожному кроці обчислювального процесу носій містить певний (поточний) кортеж значень даних, зокрема, ці значення можуть бути невизначеними.

Така модифікація моделі ЕОМ дозволила ввести Д-оператори виду $(D)O(D')$ із специфікованими даними D і D', які вони обробляють, тобто аналізують і перетворюють, змінюючи їх значення. На основі модифікованої моделі запропонована система алгоритмічних алгебр з даними (САА/Д): $< U, L, D; \Omega >$, основами якої є множина Д-операторів U, множина логічних умов L та множина даних D, а $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$ – її сигнатура, що складається з Ω_1 – операцій, що приймають значення на множині U, Ω_2 – логічних операцій, що приймають значення на множині L, Ω_3 – операцій, що приймають значення на множині даних D.

У роботі [16] розглядається варіант ІКС, коли вона складається з двох підсистем, кожна з яких функціонує у власному програмно-апаратному середовищі (двомашинний випадок). Для випадків, коли підсистеми ІКС функціонують у різних програмно-апаратних середовищах, в сигнатуру алгебри включено операцію асинхронної диз'юнкції Д-операторів $(D_i)O_i(D_i') \dot{\vee} (D_j)O_j(D_j')$, яка полягає в паралельному асинхронному виконанні цих операторів. Введені також засоби синхронізації та визначено умови, за яких є можливим паралельне виконання Д-операторів. У зв'язку з розмаїтістю архітектур ІКС і специфічністю розв'язуваних ними задач продемонстровано можливість побудови похідних засобів синхронізації.

У роботі [18] запропоновано розвиток інструментарію проектування та синтезу програм (ІПС) для конструювання специфікацій алгоритмів, що подані у вищезгаданій алгебрі алгоритмів з даними.

3. Засоби імітаційного моделювання паралельних застосувань

На даний час важлива проблема – це побудова глобальних інформаційно-

обчислювальних інфраструктур, таких як Грід-системи. Проте такі системи є складними як для створення та конфігурування, так і для програмування. При цьому підбір найбільш оптимальних конфігурацій як для структури самого Грід-середовища, так і для задач, що виконуються на ньому, недоцільно здійснювати на реальних системах через великі витрати та необхідність координації різних учасників Грід-проектів. Тому актуальною є задача імітаційного моделювання, яка дозволяє без витрат на створення, підтримку та використання реальної Грід-системи провести на моделі необхідні експерименти, результати яких не будуть суттєво відрізнятися від оригіналу. Враховуючи зростаючий інтерес до використання для обчислень графічних прискорювачів, актуальними є також задачі моделювання Грід-систем, окремі вузли яких містять CPU та GPU-компоненти.

У роботах [19–21] розроблена інструментальна система gpusim, що призначена для моделювання Грід-систем з графічними прискорювачами і грунтується на Java-фреймворку GridSim [31]. Особливістю системи є підхід до опису паралельної системи, при якому CPU і GPU-компоненти розглядаються як окремі вузли віртуального Грід-середовища. Описано механізм налаштування системи на параметри конкретної паралельної системи за рахунок автоматизованого підбору параметрів моделі.

Проведено експериментальну перевірку розробленої системи на задачах блочного множення матриць та моделювання гравітаційної взаємодії N тіл.

Для першої задачі, на основі емпіричної моделі часу виконання множення матриць на GPU, розроблені генератор експериментів симулятора і обробник статистики, що входять до складу експериментального модуля MatrixMultiply [19]. Вхідними параметрами генератора є розмір блоку, мінімальний та максимальний розмір матриці, а також інкремент розміру матриці. Результатом експерименту є залежність часу виконання множення матриць блочним алгоритмом (з урахуванням пересилання даних) від розміру матриці. Проведений експеримент показав достатню точність побудованої моделі для великих розмірів вхідних даних.

З метою використання gpusim для дослідження паралельного алгоритму обчислення гравітаційної задачі взаємодії N тіл в [20] розроблено відповідний експериментальний модуль системи gpusim. У результаті проведеного експерименту встановлено залежність часу виконання паралельної програми від розміру вхідних даних N та кількості потоків CUDA [30] на блок.

У роботі [21] розглядається паралельна розподілена реалізація системи gpusim, створена на основі використання платформи для розподілених обчислень Hazelcast [32].

4. Програмні засоби паралельних обчислень на основі платформ розподілених сховищ об’єктів даних в основній пам’яті машин

Одна основна проблема при обробці великих обсягів даних на обчислювальній техніці – це продуктивність системи введення-виведення сховищ даних. З поширенням 64-бітних багатоядерних систем вартість оперативної пам’яті значно знизилася, а обсяги багаторазово збільшилися. Це робить ідею використання оперативних запам'ятовуючих пристроїв (ОЗП) для зберігання даних замість жорстких дисків дуже привабливою, враховуючи той факт, що швидкість доступу до ОЗП у тисячі разів перевищує швидкість доступу до HDD. Об’єднання вузлів з великими обсягами ОЗП в єдиний обчислювальний кластер фактично повністю виключає потребу застосувань у доступі до повільних пристроїв введення-виведення. Усі ці чинники спричинили створення нових підходів та рішень кластеризації – розподілених сховищ об’єктів даних (In-Memory Data Grid, IMDG), які ще також називаються датагрід.

У роботі [22] виконаний аналіз та порівняння технологій сучасних платформ IMDG, а саме: RH Infinispan, Oracle Coherence, Ehcache та Hazelcast. Одне з основних застосувань IMDG-систем є клас

програм, де основна операція введення-виведення – це читання, а не запис. Прикладами таких застосувань є пошукові системи, системи агрегації та обчислень на множині даних. Проведені експерименти з виміру продуктивності кількох систем для типових застосувань на розробленому кластері з чотирьох вузлів показали, що найбільш ефективним є використання систем Coherence та Hazelcast.

У статті [23] розглядається актуальна задача автоматизованої розробки високонавантаженої розподіленої паралельної системи, яка може горизонтально масштабуватись та забезпечувати безперебійну обробку потокових даних великих обсягів. Як приклад джерела даних для обробки обрана соціальна мережа Twitter та її потоковий API – Twitter Firehose. Для створення розподіленої системи використано Hazelcast [32], перевагою якого є автоматичне розгортання та керування обчислювальним кластером. Для розгортання системи на основі Hazelcast обрана "хмарна" платформа Amazon Elastic Compute Cloud (EC2). Розроблена система є динамічно масштабованим та відмовостійким кластерним рішенням, одним з головних надбань якого є те, що дослідник може нарощувати ресурси для обчислень за своїм бажанням, різної конфігурації та потужності, що дає змогу досягати бажаної швидкодії системи.

5. Засоби розробки адаптивних алгоритмів на основі параметрично керованої генерації схем програм

Одним із шляхів вирішення проблеми підвищення адаптивності програм до конкретних умов їх використання у рамках алгеброалгоритмічного підходу є застосування параметрично керованої генерації високорівневих специфікацій алгоритмів за допомогою схем більш високого рівня, що називаються гіперсхемами.

У роботі [15] запропоновано підхід до розробки послідовних і паралельних алгоритмів на основі алгебри гіперсхем та інструментальних засобів генерації САА-схем алгоритмів, що входять до складу інтегрованого інструментарію проектування та синтезу програм (див. також підрозділ 1.2). На рис. 1 показана послідовність розробки програм у системі ІПС, починаючи з проектування гіперсхеми та закінчуючи генерацією коду цільовою мовою програмування.

Гіперсхеми є параметризованими операторними виразами, що орієнтовані на вирішення певного класу задач. Зазначення конкретних значень параметрів гіперсхем та їх наступна інтерпретація дозволяє отримати САА-схеми алгоритмів, адаптовані до конкретних умов використання. У роботі [15] застосування гіперсхем проілюстроване на прикладах з області сортування та лінійної алгебри.

Приклад 1. Далі наведено гіперсхему, яка є алгоритмом керування виведенням класу САА-схем паралельного множення матриць із різною кількістю паралельних потоків K. У САА-схемі, яка є результатом інтерпретації гіперсхеми, виконується множення матриці $A=(a_{lj})_{M\times N}$ на матрицю $B=(b_{lj})_{N\times Q}$. Матриця A розділена на K горизонтальних смуг (блоків). Обчислення розподіляються між K паралельними потоками з індексами $i=0,\ldots,K-1$. При цьому i-й потік, а саме, оператор "Множення($A(i)$, B)", виконує множення i-го блоку матриці A на матрицю B. Параметра K, який відповідає кількості паралельних потоків, у гіперсхемі присвоєне значення 4.

```
"Гіперсхема множення матриць" =
==== (i := –1);
     (K := 4);
     "ПРС1";
     ЧЕКАТИ 'Обробка у всіх
                потоках закінчена';

"ПРС1" = "ІНК(i)";
         ВИБІР
         (
           ['i < K – 1'] →
             "Множення(A(i), B)"
             ПАРАЛЕЛЬНО
             "ПРС1";

           ['i = K – 1'] →
             "Множення(A(i), B)";
         ).
```

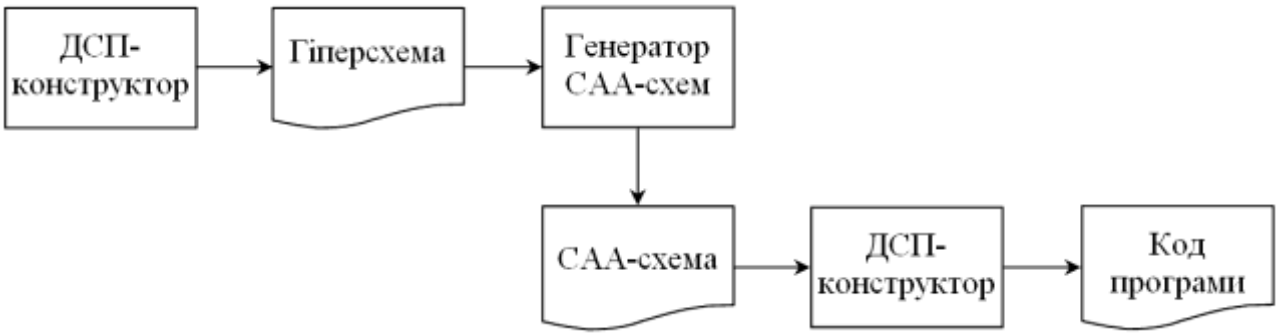

Рис. 1. Послідовність генерації алгоритмів та програм в інтегрованому інструментарії

У вищенаведеній гіперсхемі складений оператор ПРС1 рекурсивно формує паралельні потоки із використанням операції вибору та зміни значення параметра i від 0 до $K-1$. Результатом інтерпретації гіперсхеми є САА-схема множення матриць з чотирма паралельними потоками.

6. Засоби автоматичного налаштовування застосувань на мультипроцесорні платформи

У роботах [24, 25] розроблено метод автоматичного самоналаштовування (автотюнінгу) паралельних застосувань на основі формальної моделі розширеного поняття дискретної динамічної системи та PRAM-моделі паралельних обчислень. Автотюнінг дозволяє емпіричним способом в автоматичному режимі підібрати найкращий варіант програми для цільового середовища виконання (мультипроцесорної платформи). На основі запропонованого методу розроблено інструментальну систему генерації автотюнерів TuningGenie.

На рис. 2 показано процес модифікації, оцінки і вибору оптимального варіанта програми в TuningGenie. Система сприймає на вході початковий код мовою програмування, відмічений прагмами. Прагми описують конфігурації і трансформації програми, що впливають на її продуктивність. Прагми задаються вручну програмними розробниками із використанням коментарів спеціальної форми. Спочатку інформація з усіх прагм у коді вхідної програми збирається синтаксичним аналізатором і на її основі генеруються усі можливі конфігурації застосунку. Далі для кожної конфігурації генерується відповідна варіація програми й виконуються заміри швидкодії. На основі отриманих результатів знаходиться оптимальна конфігурація і на її основі генерується оптимальний варіант застосунка. Для трансформацій вихідного коду TuningGenie використовує інструментарій переписувальних правил TermWare [8, 9, 25].

Застосування TuningGenie продемонстроване у роботах [24–26] на прикладах задач паралельного програмування, зокрема, задачі короткочасного метеорологічного прогнозування.

7. Прикладні програмні системи метеорологічного прогнозування

У роботах [11–13, 27, 28] виконане застосування методології та інструментарію автоматизованого алгеброалгоритмічного проектування (див. розділ 1) для розробки паралельних алгоритмів та програмної реалізації розв'язання задач регіонального метеорологічного прогнозування. Запропонований підхід до вирішення проблем метеорологічного прогнозування поєднує комплексність використання адекватних фізичних моделей атмосферних процесів з ефективними обчислювальними методами програмування високопродуктивних обчислень на багатоядерних процесорах та графічних прискорювачах, що дає змогу досягати належного ступеня точності, повноти і своєчасності інформації, необхідної для складання якісних метеорологічних прогнозів.

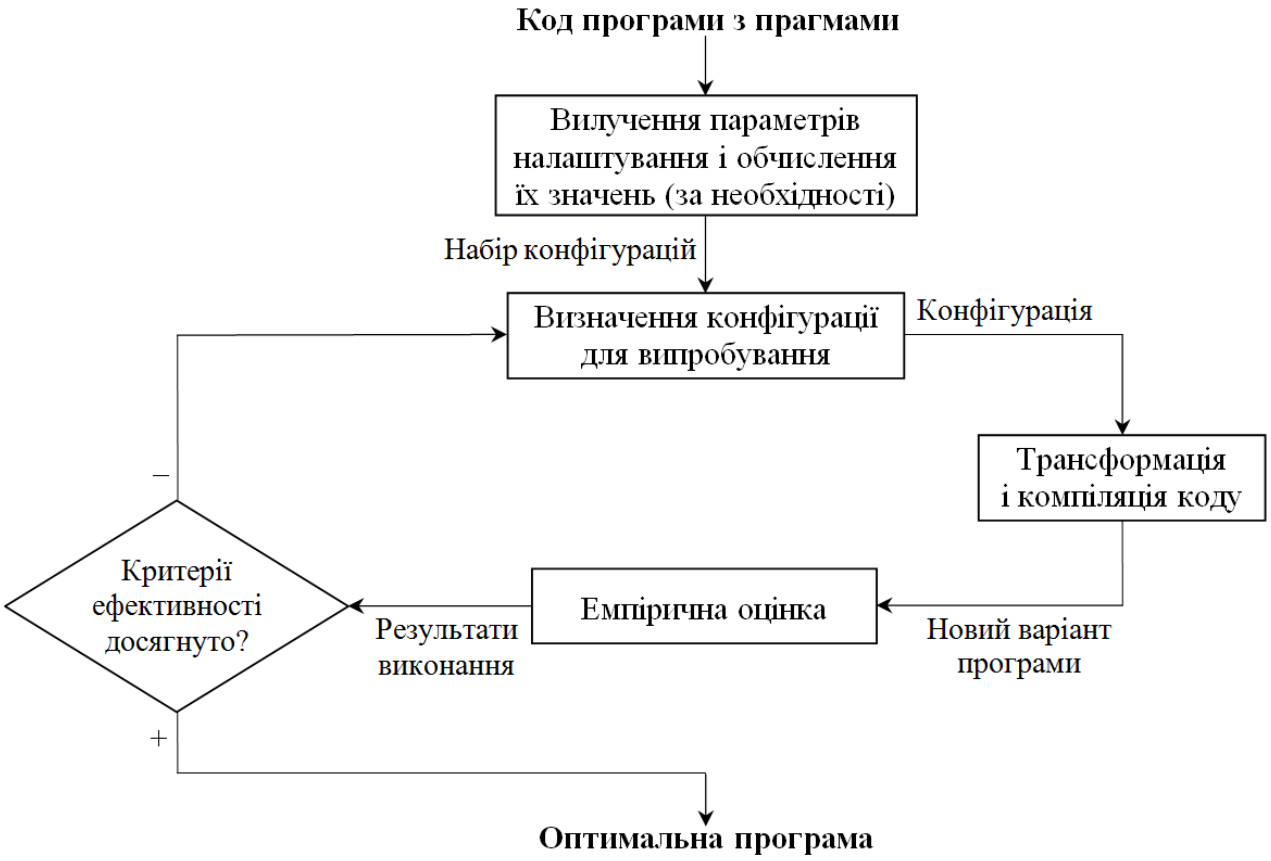

Рис. 2. Послідовність дій автотюнінгу в системі TuningGenie

Приклад 2. Далі наведено частину паралельної САА-схеми обчислень прогнозу погоди, в основу якої покладено регіональну математичну модель стану атмосфери та чисельні методи, що розглядаються в роботі [11].

"Паралельне обчислення правих частин рівнянь *Q* для функції (*U*)" =
====
Запуск *Ядра* (*blocksPerGrid_3d*, *threadsPerBlock_3d*)
("Виконати попередні обчислення для першої похідної (*Pr1*)");
Запуск *Ядра* (*blocksPerGrid_1d*, *threadsPerBlock_1d*)
("Виконати обчислення першої похідної (*Pr1*)");
"Заповнити масив для *Q* у відеопам'яті значеннями (0)";
Запуск *Ядра* (*blocksPerGrid_3d*, *threadsPerBlock_3d*)
("Обчислити праві частини *Q* для функції (*U*)");
"Занести в масив (*RZ*) значення для функції (*U*)";
"Обчислити складові рівняння переносу в напрямках X,Y,Z для (*U*)";
"Скопіювати масив значень *Q* з відеопам'яті у пам'ять CPU".

На основі побудованої САА-схеми реалізовано послідовну та паралельну програми мовою C++ для виконання на CPU та GPU відповідно [11]. Випробування програм проводились із використанням процесора i5-3570 та графічного прискорювача GeForce GTX 650 Ti (768 ядра). На рис. 3 показано графік залежності мультипроцесорного прискорення $Sp = T_s / T_p$ (де T_s і T_p – час виконання послідовної та паралельної програм, відповідно), від розміру задачі N – кількості вузлів розбиття розрахункової сітки.

У роботі [27] запропоновано підхід до проектування та генерації Грід-сервісів для платформи Globus Toolkit на основі використання інструментарію ІПС. Розро-

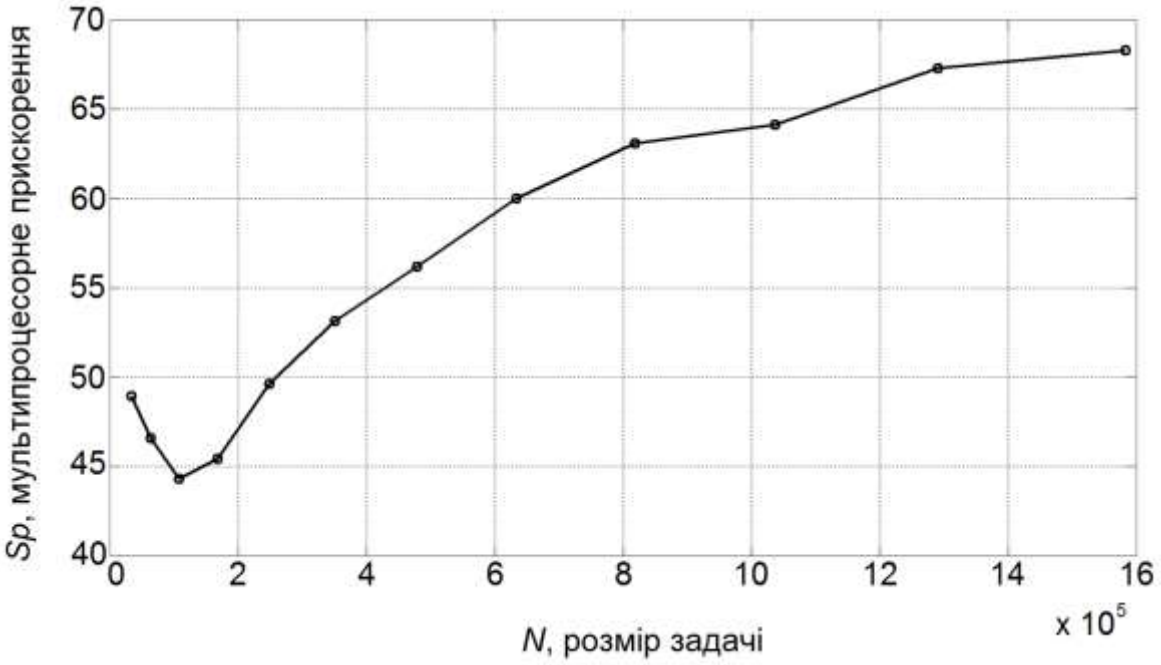

Рис. 3. Графік залежності мультипроцесорного прискорення від розміру задачі для паралельної програми метеорологічного прогнозування

блено Грід-сервіс, що виконує запуск паралельної програми з області метеорологічного прогнозування на мультипроцесорному кластері.

В роботі [28] розроблено засоби автоматизованого конструювання паралельного коду для середовища OpenMP на основі високорівневих алгеброалгоритмічних специфікацій. Застосування засобу демонструється на прикладі задачі моделювання циркуляції атмосфери, що представлений як сервіс у складі Інтернет-порталу [29] з надання послуг метеопрогнозу.

Висновки

В статті показано, як в Інституті програмних систем НАНУ розвивалися формальні та адаптивні методи та інструментарій для автоматизованого проектування та генерації паралельних програм на основі алгеброалгоритмічного підходу та техніки переписувальних правил. За роки становлення і розвитку цього напрямку розроблено нові засоби опису паралельних алгоритмів у рамках алгебри алгоритмів з даними для класу інформаційно-керуючих систем. Створені інструментальні засоби автоматизації програмування та імітаційного моделювання високопродуктивних паралельних застосувань для Грід-систем. Розроблена кластерна система для обробки потокових даних великого обсягу на основі платформи, призначеної для створення розподілених сховищ об'єктів даних в основній пам'яті машин. Запропоновано метод та програмні засоби розробки послідовних і паралельних адаптивних алгоритмів на основі використання параметрично керованої генерації схем програм. Розроблено метод та програмну систему для автоматичного самоналаштовування (автотюнінгу) паралельних програм на цільові платформи, що ґрунтується на трансформації програмного коду на основі використання переписувальних правил. Проведено застосування методології автоматизованого алгеброалгоритмічного проектування для розробки паралельних алгоритмів та програмної реалізації розв'язання задач регіонального метеорологічного прогнозування на різних мультипроцесорних платформах, що дало можливість значно покращити точність та завчасність прогнозів.

1. Глушков В.М. Теория автоматов и формальные преобразования микропрограмм. *Кибернетика*. 1965. № 5. С. 1–10.

2. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Методы символьной мультиобработки. К.: Наукова думка, 1980. 252 с.
3. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. 3-е изд., перераб. и доп. К.: Наукова думка, 1989. 376 с.
4. Капитонова Ю.В., Летичевский А.А. Парадигмы и идеи академика В. М. Глушкова. К.: Наукова думка, 2003. 456 с.
5. Капитонова Ю.В., Летичевский А.А. Методы и средства алгебраического программирования. *Кибернетика*. 1993. № 3. С. 7–12.
6. Letichevsky A.A., Kapitonova Yu.V., Konozenko S.V. Computations in APS. *Theoretical computer science*. 1993. Vol. 119. P. 145–171.
7. Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А. Алгеброалгоритмические модели и методы параллельного программирования. Киев: Академпериодика, 2007. 631 с.
8. Андон Ф.И., Дорошенко А.Е., Жереб К.А., Шевченко Р.С., Яценко Е.А. Методы алгебраического программирования. Формальные методы разработки параллельных программ. Киев: Наукова думка, 2017. 440 с.
9. Doroshenko A., Shevchenko R. A rewriting framework for rule-based programming dynamic applications. *Fundamenta Informaticae*. 2006. Vol. 72, N 1–3. P. 95–108.
10. Андон Ф.И., Дорошенко А.Е., Жереб К.А. Программирование высокопроизводительных параллельных вычислений: формальные модели и графические ускорители. *Кибернетика и системный анализ*. 2011. № 4. С. 176–187.
11. Дорошенко А.Ю., Бекетов О.Г., Прусов В.А., Тирчак Ю.М., Яценко О.А. Формалізоване проектування та генерація паралельної програми чисельного моделювання погоди. *Проблеми програмування.* 2014. № 2–3. С. 72–81.
12. Андон Ф.И., Дорошенко А.Е., Бекетов А.Г., Иовчев В.А., Яценко Е.А. Инструментальные средства автоматизации параллельного программирования на основе алгебры алгоритмов. *Кибернетика и системный анализ*. 2015. № 1. С. 162–170.
13. Дорошенко А.Ю., Бекетов О.Г., Іванів Р.Б., Іовчев В.О., Мироненко І.О., Яценко О.А. Автоматизована генерація паралельних програм для графічних прискорювачів на основі схем алгоритмів. *Проблеми програмування*. 2015. № 1. С. 19–28.
14. Doroshenko A., Zhereb K., Yatsenko O. Developing and optimizing parallel programs with algebra-algorithmic and term rewriting tools. *Proc. 9th International Conference "ICT in Education, Research, and Industrial Applications"* (ICTERI 2013), Revised Selected Papers, Kherson, Ukraine (19–22 June, 2013). Berlin: Springer, 2013. Vol. 412. P. 70–92.
15. Яценко Е.А. Средства параметрически управляемой генерации алгоритмов на основе алгебры гиперсхем. *Проблеми програмування*. 2012. № 2–3. С. 219–227.
16. Акуловский В.Г., Дорошенко А.Е. Описание параллелизма в алгоритмах информационно-управляющих систем средствами алгебраического аппарата. *Проблеми програмування*. 2013. № 3. С. 13–21.
17. Акуловский В.Г., Дорошенко А.Е. Преобразование алгоритмов, записанных в виде композиционных схем. *Кибернетика и системный анализ*. 2014. № 1. С. 151–159.
18. Акуловский В.Г., Дорошенко А.Е., Яценко Е.А. Реализация средств проектирования и генерации программ на основе алгебры алгоритмов с данными. *Проблеми програмування*. 2015. № 2. С. 41–51.
19. Оконський І.В., Дорошенко А.Ю., Жереб К.А. Інструментальні засоби моделювання гетерогенних середовищ заснованих на відеографічних прискорювачах. *Проблеми програмування*. 2013. № 1. С. 107–115.
20. Дорошенко А.Ю., Оконський І.В., Жереб К.А., Бекетов О.Г. Використання засобів моделювання для визначення оптимальних параметрів виконання програм на відеографічних прискорювачах. *Проблеми програмування*. 2013. № 2. С. 23–31.
21. Дорошенко А.Ю., Гнинюк М.В. Паралельна розподілена реалізація моделювання паралельних обчислень. *Проблеми програмування*. 2014. № 1. С. 40–48.
22. Рухлис К.А., Дорошенко А.Е. К вопросу о производительности распределенных хранилищ объектов данных в памяти ОЗУ. *Проблеми програмування*. 2015. № 3. С. 33–38.
23. Тітов Д.С., Дорошенко А.Ю., Яценко О.А. Автоматизована розробка паралельної розподіленої системи обробки потокових даних. *Проблеми програмування*. 2016. № 2–3. С. 96–104.
24. Иваненко П.А., Дорошенко А.Е. Метод автоматической генерации автотьюнеров для параллельных программ. *Кибернетика и системный анализ*. 2014. № 3. С. 161–173.

25. Ivanenko P., Doroshenko A., Zhereb K. TuningGenie: Auto-tuning framework based on rewriting rules. *Proc. 10th International Conference "ICT in Education, Research, and Industrial Applications"* (ICTERI 2014), Revised Selected Papers, Kherson, Ukraine (9–12 June, 2014). Berlin: Springer, 2014. Vol. 469. P. 139–158.
26. Іваненко П.А., Дорошенко А.Ю. Автоматична оптимізація виконання для задачі метеорологічного прогнозування. *Проблеми програмування*. 2012. № 2–3. С. 426–434.
27. Дорошенко А.Ю., Бекетов О.Г., Яценко О.А, Вітряк Є.А., Павлючин Т.О. Розробка сервісно-орієнтованих засобів для запуску паралельних програм на мультипроцесорному кластері. *Проблеми програмування*. 2014. № 4. С. 3–14.
28. Дорошенко А.Ю., Іваненко П.А., Овдій О.М., Яценко О.А. Автоматизоване проектування програм для розв'язання задачі метеорологічного прогнозування. *Проблеми програмування*. 2016. № 1. С. 102–115.
29. Бекетов О.Г., Вітряк Є.А., Мироненко І.О., Овдій О.М. Розвиток Інтернет-порталу метеорологічного прогнозування на мультипроцесорній платформі. *Проблеми програмування*. 2016. № 2–3. С. 246–253.
30. Wilt N. The CUDA handbook. A comprehensive guide to GPU programming. Boston: Addison-Wesley, 2013. 528 p.
31. Sulistio A., Cibej U., Venugopal S., Robic B., Buyya R. A toolkit for modelling and simulating data Grids: an extension to GridSim. *Concurrency and Computation: Practice & Experience*. 2008. Vol. 20, N 13. P. 1591–1609.
32. Veentjer P. Mastering Hazelcast: the ultimate Hazelcast book. Palo Alto: Hazelcast, 2015. 232 p.

References

1. Glushkov V.M. (1965) Automata theory and structural design problems of digital machines. Cybernetics. (5). P. 1–10. (in Russian).
2. Glushkov V.M., Tseitlin G.E. & Yushchenko E.L. (1980) Methods of symbolic multiprocessing. Kyiv: Naukova dumka. (in Russian).
3. Glushkov V.M., Tseitlin G.E. & Yushchenko E.L. (1989) Algebra. Languages. Programming. 3rd edition. Kyiv: Naukova dumka. (in Russian).
4. Kapitonova Yu.V. & Letichevsky A.A. (2003) Paradigms and ideas of academician V.M. Glushkov. Kyiv: Naukova dumka. (in Russian).
5. Kapitonova Yu.V. & Letichevsky A.A. (1993) Algebraic programming: methods and tools. Cybernetics. (3). P. 7–12. (in Russian).
6. Letichevsky A.A., Kapitonova Yu.V. & Konozenko S.V. (1993) Computations in APS. Theoretical computer science. 119. P. 145–171.
7. Andon P.I. et al. (2007) Algebra-algorithmic models and methods of parallel programming. Kyiv: Akademperiodika. (in Russian).
8. Andon P.I. et al. (2017) Methods of algebraic programming. Formal methods of parallel program development. Kyiv: Naukova dumka. (in Russian).
9. Doroshenko A. & Shevchenko R. (2006) A rewriting framework for rule-based programming dynamic applications. Fundamenta Informaticae. 72 (1–3). P. 95–108.
10. Andon P.I., Doroshenko A.Yu. & Zhereb K.A. (2011) Programming high-performance parallel computations: formal models and graphics processing units. Cybernetics and Systems Analysis. (4). P. 176–187. (in Russian).
11. Doroshenko A.Yu., Beketov O.G., Prusov V.A., Tyrchak Yu.M. & Yatsenko O.A. (2014) Formalized designing and generation of parallel program for numerical weather forecasting task. Problems in programming. (2–3). P. 72–81. (in Ukrainian).
12. Andon P.I., Doroshenko A.Yu., Beketov O.G., Iovchev V.O. & Yatsenko O.A. (2015) Software tools for automation of parallel programming on the basis of algebra of algorithms. Cybernetics and systems analysis. (1). P. 162–170. (in Russian).
13. Doroshenko A.Yu., Beketov O.G., Ivaniv R.B., Iovchev V.O., Myronenko I.O. & Yatsenko O.A. (2015) Automated generation of parallel programs for graphics processing units based on algorithm schemes. Problems in programming. (1). P. 19–28. (in Ukrainian).
14. Doroshenko A., Zhereb K. & Yatsenko O. (2013) Developing and optimizing parallel programs with algebra-algorithmic and term rewriting tools. In Proc. 9th International Conference "ICT in Education, Research, and Industrial Applications" (ICTERI 2013), Revised Selected Papers. Kherson, Ukraine, 19–22 June 2013. Berlin: Springer. 412. P. 70–92.

15. Yatsenko O.A. (2012) Facilities for parameter-driven generation of algorithms on the basis algebra of hyperschemes. Problems in programming. (2–3). P. 219–227. (in Russian).
16. Akulovskiy V.G. & Doroshenko A.Yu. (2013) Description of parallelism in the algorithms of the information/control systems with algebraic facilities. Problems in programming. (3). P. 13–21. (in Russian).
17. Akulovskiy V.G. & Doroshenko A.Yu. (2014) Transformation of algorithms written in the form of composition schemes. Cybernetics and Systems Analysis. (1). P. 151–159. (in Russian).
18. Akulovskiy V.G., Doroshenko A.Yu. & Yatsenko O.A. (2015) Implementation of tools for designing and generating of programs on the basis of algebra of algorithms with data. Problems in programming. (2). P. 41–51. (in Russian).
19. Okonsky I.V., Doroshenko A.Yu. & Zhereb K.A. (2013) Instrumental means of simulation of heterogeneous environments based on graphics processing units. Problems in programming. (1). P. 107–115. (in Ukrainian).
20. Doroshenko A.Yu., Okonsky I.V., Zhereb K.A. & Beketov O.G. (2013) Using means of simulation for determining optimal parameters for running programs on GPU. Problems in Programming. (2). P. 23–31. (in Ukrainian).
21. Doroshenko A.Yu. & Gnynjuk M.V. (2014) Parallel distributed implementation of simulation of parallel computation. Problems in programming. (1). P. 40–48. (in Ukrainian).
22. Rukhlis K.A. & Doroshenko A.Yu. (2015) On the performance of the in-memory data grids. Problems in programming. (3). P. 33–38. (in Russian).
23. Titov D.S., Doroshenko A.Yu. & Yatsenko O.A. (2016) Automated development of a parallel system for distributed streaming data processing. Problems in programming. (2-3). P. 96–104. (in Ukrainian).
24. Ivanenko P.A. & Doroshenko A.Yu. (2014) Method of automated generation of autotuners for parallel programs. Cybernetics and systems analysis. (3). P. 161–173. (in Russian).
25. Ivanenko P., Doroshenko A. & Zhereb K. (2014) TuningGenie: auto-tuning framework based on rewriting rules. In Proc. 10th International Conference "ICT in Education, Research, and Industrial Applications" (ICTERI 2014), Revised Selected Papers. Kherson, Ukraine, 9–12 June 2014. Berlin: Springer. 469. P. 139–158.
26. Ivanenko P.A. & Doroshenko A.Yu. (2012) Automatic optimization of execution for the meteorological forecasting problem. Problems in programming. (2–3). P. 426–434. (in Ukrainian).
27. Doroshenko A.Yu., Beketov O.G. Yatsenko O.A., Pavliuchyn T.O. & Vitriak Ie.A. (2014) Development of the service-oriented software for launching parallel programs on a multiprocessor cluster. Problems in programming. (4). P. 3–14. (in Ukrainian).
28. Doroshenko A.Yu., Ivanenko P.A., Ovdii O.M. & Yatsenko O.A. (2016) Automated program design for solution of weather forecasting problem. Problems in programming. (1). P. 102–115. (in Ukrainian).
29. Beketov O.G., Vitriak Ie.A., Myronenko I.O. & Ovdii O.M. (2016) Development of meteorological forecasting web portal on multiprocessor platform. Problems in programming. (2-3). P. 246–253. (in Ukrainian).
30. Wilt N. (2013) The CUDA handbook. A comprehensive guide to GPU programming. Boston: Addison-Wesley.
31. Sulistio A., Cibej U., Venugopal S., Robic B. & Buyya R. (2008) A toolkit for modelling and simulating data Grids: an extension to GridSim. Concurrency and Computation: Practice & Experience. 20 (13). P. 1591–1609.
32. Veentjer P. (2015) Mastering Hazelcast: the ultimate Hazelcast book. Palo Alto: Hazelcast.

Одержано 12.06.2017

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень Інституту
програмних систем НАН України,
професор кафедри автоматики
та управління в технічних системах
НТУ України "КПІ".

Кількість наукових публікацій в
українських виданнях – понад 200.
Кількість наукових публікацій в
зарубіжних виданнях – понад 50.
Індекс Хірша – 5.
http://orcid.org/0000-0002-8435-1451.

Бекетов Олексій Геннадійович,
молодший науковий співробітник
Інституту програмних
систем НАН України.
Кількість наукових публікацій в
українських виданнях – 10.
Кількість наукових публікацій в
зарубіжних виданнях – 1.
http://orcid.org/0000-0003-4715-5053.

Жереб Костянтин Анатолійович,
кандидат фізико-математичних наук,
старший науковий співробітник Інституту
програмних систем НАН України.
Кількість наукових публікацій в
українських виданнях – понад 30.
Кількість наукових публікацій в
зарубіжних виданнях – понад 10.
http://orcid.org/0000-0003-0881-2284.

Іваненко Павло Андрійович,
молодший науковий співробітник
Інституту програмних
систем НАН України.
Кількість наукових публікацій в
українських виданнях – 16.
Кількість наукових публікацій в
зарубіжних виданнях – 4.
http://orcid.org/0000-0001-5437-9763.

Овдій Ольга Михайлівна,
молодший науковий співробітник
Інституту програмних
систем НАН України.
Кількість наукових публікацій в
українських виданнях – 20.
Кількість наукових публікацій в
зарубіжних виданнях – 4.
http://orcid.org/0000-0002-8891-7002.

Шевченко Руслан Сергійович,
директор ТОВ "Град-Софт".
Кількість наукових публікацій в
українських виданнях – 7.
Кількість наукових публікацій в
зарубіжних виданнях – 9.
http://orcid.org/0000-0002-1554-2019.

Яценко Олена Анатоліївна,
кандидат фізико-математичних наук,
старший науковий співробітник Інституту
програмних систем НАН України.
Кількість наукових публікацій в
українських виданнях – 35.
Кількість наукових публікацій в
зарубіжних виданнях – 12.
http://orcid.org/0000-0002-4700-6704.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.

ТОВ "Град-Софт",
02068, Київ-068,
вул. Ревуцького, 29, к. 240.
Тел.: (044) 552 8259.

E-mail: doroshenkoanatoliy2@gmail.com,
beketov.oleksii@gmail.com,
zhereb@gmail.com,
paiv@ukr.net,
olga.ovdiy@gmail.com,
ruslan.s.shevchenko@gmail.com,
oayat@ukr.net.