КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 004.89 DOI:10.34229/2707-451X.20.1.8

В.Г. ТУЛЬЧИНСЬКИЙ, С.І. ЛАВРЕНЮК, В.Ю. РОГАНОВ,
П.Г. ТУЛЬЧИНСЬКИЙ, В.В. ХАЛІМЕНДІК

ФАКТОРИ ПРОДУКТИВНОСТІ ЗАСТОСУВАННЯ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ У ХМАРІ З ВИКОРИСТАННЯМ GPU

Експериментально досліджені фактори, що впливають на продуктивність застосування готових нейромережних моделей у хмарних системах різної архітектури з графічними прискорювачами. Оцінені накладні видатки пов'язані з мікросервісною і розподіленою архітектурою, вплив пам'яті, мережі, розміру пакетів, синхронної і асинхронної взаємодії. Продемонстровано складний нелінійний характер впливу параметрів системи у різних комбінаціях.

Ключові слова: *машинне навчання, хмарні технології, графічні прискорювачі, GPU, системна архітектура, продуктивність.*

© В.Г. Тульчинський, С.І. Лавренюк, В.Ю. Роганов, П.Г. Тульчинський, В.В. Халімендік, 2020

Вступ. У роботах з машинного навчання (МН) і штучного інтелекту (ШІ) наголос зазвичай робиться на якості класифікації, або точності оцінки параметрів. Якщо увага приділяється продуктивності, то переважно також йдеться про продуктивність етапу тренування моделі. Проте з розповсюдженням застосувань ШІ в реальних прикладних задачах важливішою стає проблема забезпечення високої продуктивності обробки даних з допомогою готових натренованих моделей. За своїм характером ця проблема принципово відрізняється від проблеми тренування моделей: остання має справу з інтенсивними обчисленнями, а перша – з простими обчисленнями, але великими потоками даних (файлів), що надходять з мережі чи файлової системи на обробку. Тобто це типова задача паралельної обробки з інтенсивним введенням/виведенням.

Додамо, що з точки зору прикладного застосування, модуль ШІ, що виконує класифікацію, оцінку, чи іншу обробку даних є «чорним ящиком»: вартість розробки і тренування моделі, як і ризики невдачі занадто високі, щоб займатися такими задачами не професійно. Тому оптимізація продуктивності насамперед передбачає підбор та балансування параметрів середовища. Хмарні системи з їх гнучкістю, керованістю і легким масштабуванням є ідеальними площадками для таких задач.

Розглянемо детальніше задачу дослідження факторів, що впливають на продуктивність на одному, але визначному прикладі.

Історію розпочала організація ImageNet [1], яка веде відкриту базу даних посилань на зображення з Інтернету, організовану у семантичну ієрархію WordNet за іменниками. В ній кожен вузол ієрархії відповідає певному слову та містить посилання на сотні чи тисячі відповідних зображень. У середньому – 500 зображень на вузол. Спочатку ImageNet був асоційований з Стенфордським університетом,

потім переїхав в Університет Північної Кароліни в Чапел Хілл. Починаючи з 2010 р. ImageNet проводить конкурс автоматичного розпізнавання зображень Large Scale Visual Recognition Challenge (ILSVRC), в якому беруть участь провідні компанії (як Google і Microsoft), університети, дослідницькі організації тощо. У 2015 р. цей конкурс виграла група з Microsoft Research Asia – Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, що запропонувала нову нейронну мережу глибокого навчання за залишками, а саме її варіант з 50 шарами: ResNet-50 [2]. Наступні експерименти були навіть успішніші: ResNet з 152 шарами досягла 3% рівня помилок, що навіть краще за людину [3]. Та завдяки успіху 2015 р. комбінація ImageNet / ResNet-50 стала одна з найпопулярніших для тестування широкомасштабного розподіленого глибокого навчання [4]. Для тестування продуктивності розпізнавання ми використовували ResNet-50 з вибраними даними з колекції ImageNet.

Експерименти. Експериментально досліджено три архітектури системи розпізнавання на основі ResNet-50, які показано на рис. 1 – 3. Для реалізації програм використано мову Python.

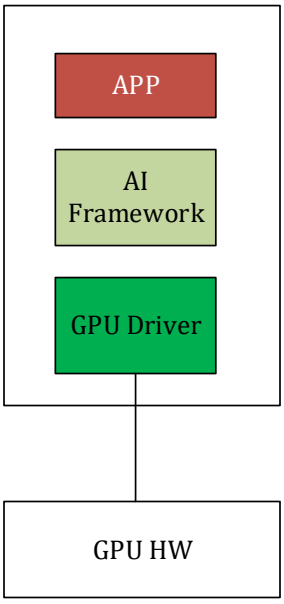

РИС. 1. Проста архітектура

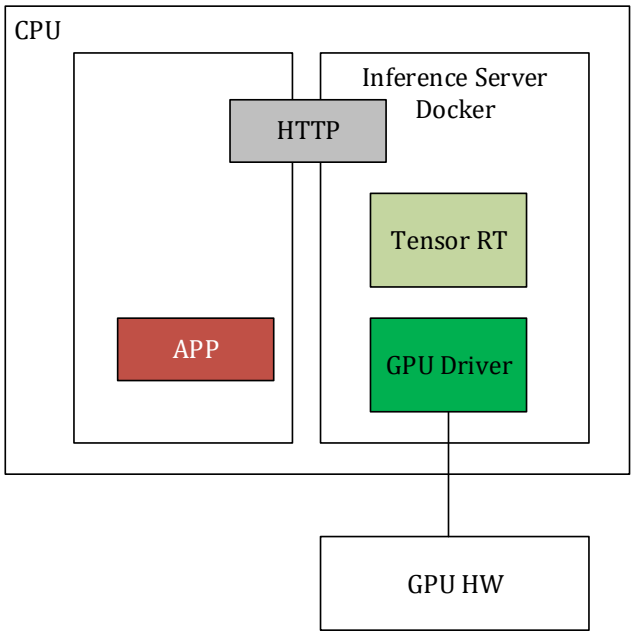

РИС. 2. Мікросервісна архітектура

На рис. 1 – проста архітектура, де система ШІ (AI Framework) не відокремлена від програми користувача (APP), а поділяє з нею адресний простір спільної віртуальної машини (ВМ). Програма користувача завантажує дані (тобто картинки), об'єднує їх у пакети та подає на бібліотеку ШІ TensorFlow. ШІ через драйвер NVIDIA (GPU Driver) використовує графічний прискорювач (GPU HW), що через механізм портів прив'язаний до ВМ. Програма працює включно у синхронному режимі, тому що бібліотечні виклики TensorFlow не розпаралелюються.

На рис. 2 – мікросервісна архітектура, що використовує створений та оптимізований NVIDIA спеціалізований Docker-контейнер (Inference Server Docker) з програмою ШІ TensorRT замість бібліотеки TensorFlow. Програма користувача (APP) виконується в пам'яті хоста (ВМ), а для передачі пакетів у контейнер використовуються мережеві протоколи HTTP або GRPC (прискорений протокол від Amazon). Це дозволяє обирати між синхронним і асинхронним режимом подачі пакетів на обробку.

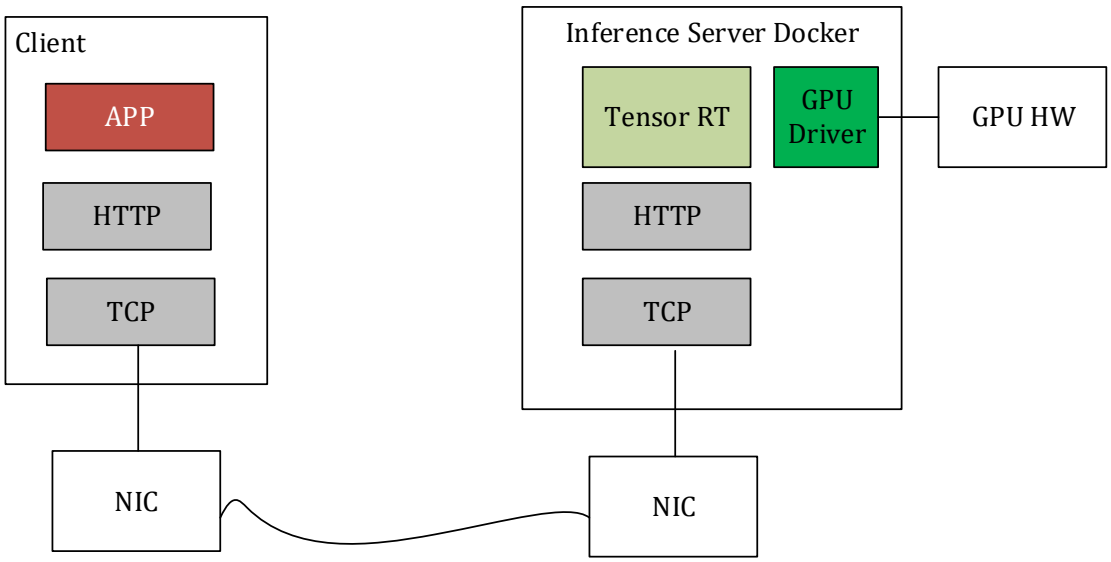

РИС. 3. Розподілена архітектура

На рис. 3 – розподілена архітектура, в якій програма користувача (APP) та Inference Server Docker виконуються на різних вузлах. Відповідно програма користувача залишає весь процесорний ресурс і пам'ять хоста у розпорядженні контейнера ШІ, але мережевий канал використовується вже не для копіювання даних у пам'яті одного фізичного вузла, а для дійсної передачі пакетів з вузла на вузол (також через мережеві протоколи HTTP або GRPC) з допомогою мережевих карт (NIC) та мережевих протоколів нижчого рівня, зокрема, стеку TCP для HTTP. Ця архітектура також дозволяє вибір між синхронним і асинхронним режимом подачі пакетів на обробку.

Тестовий набір даних для експериментального дослідження продуктивності розпізнавання з допомогою моделі ResNet-50 був завантажений за посиланнями ImageNet з колекцій №№ n12154773, n04154340, n04516672, n02729837, n04105893, n07881800, n03800933. Оскільки ImageNet містить не файли картинок, а лише Інтернет-посилання, не всі вони дозволяють завантажити файл. Також, було з'ясовано, що не всі завантажені картинки розпізнаються моделлю ResNet-50. Зокрема, з'ясувалося що картинки менше за 2052 байт не розпізнаються зовсім. Деякі більші за розміром файлу картинки фактично виявились теж недосить великими для успішного розпізнавання. Оскільки задача полягала виключно в оцінці параметрів продуктивності, тестовий набір був заздалегідь програмно перевірений на можливість розпізнавання, і зображення, що не були розпізнані, відбраковувалися. Також з'ясувалося, що розмір файлу в пікселях впливає на продуктивність розпізнавання через додаткове перетворення до стандартного розміру. Щоб виключити цей фактор з урахування, весь набір картинок був перед вимірюваннями приведений до стандартного розміру 224x224 відповідно до розмірів вхідного шару моделі ResNet-50. Тестова колекція складалася з 1500 різних картинок. Але для перевірки стабільної швидкодії і стійкості файли були розмножені, і додаткові експерименти проводилися з набором 13500 картинок. Вони підтвердили надійність оцінок отриманих для 1500 файлів.

Під час розпізнавання тестовий набір даних знаходився у локальній пам'яті хоста, на якому виконувалася програма APP (в розподіленій архітектурі – на клієнті, в інших – на сервері).

Технічні характеристики обладнання, що використовувалось для експериментального дослідження продуктивності розпізнавання та факторів, що на неї впливають, зведені у табл. 1.

ТАБЛИЦЯ 1. Технічні характеристики тестового устаткування

Сервер (хост)			**Клієнт (для розподіленої архітектури)**		
Вузол:	p3.2xlarge		Вузол:	t3.2xlarge	
ЦПУ:	Intel Xeon E5-2686 v4		ЦПУ:	Intel Scalable	
Кількість ядер:	8		Кількість ядер:	4	
Тактова частота:	2.3	ГГц	Тактова частота:	2.5	ГГц
Пам'ять (ОЗУ):	61	ГБ	Пам'ять (ОЗУ):	16	ГБ
GPU:	NVIDIA V100		**Мережа (для розподіленої архітектури)**		
Пам'ять GPU:	16	ГБ	Швидкість:	5	Гбіт/с

Результати. Залежність загальної продуктивності (кількості оброблених картинок на секунду) від архітектури і розміру пакету показана на рис. 4.

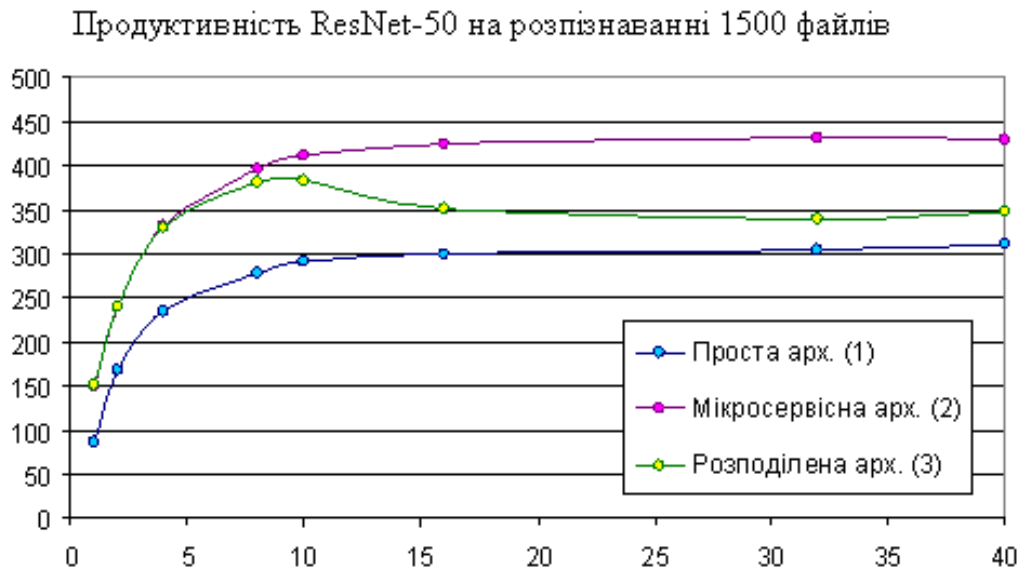

РИС. 4. Продуктивність ResNet-50 (асинхронний режим, GRPC)

Для мікросервісної і розподіленої архітектури вимірювання проводилися в асинхронному режимі з мережевим протоколом GRPC, для простої – у синхронному режимі з безпосереднім викликом функції розпізнавання. Переваги мікросервісної (2) і розподіленої архітектури (3) над простою (1) частково пояснюється підтримкою асинхронного завантаження файлів, а частково – кращою оптимізацією TensorRT в Inference Server Docker у порівнянні з універсальною бібліотекою TensorFlow. Більш детальне порівняння трьох архітектур у різних режимах показано на рис. 5 дозволяє краще оцінити вагу тих факторів.

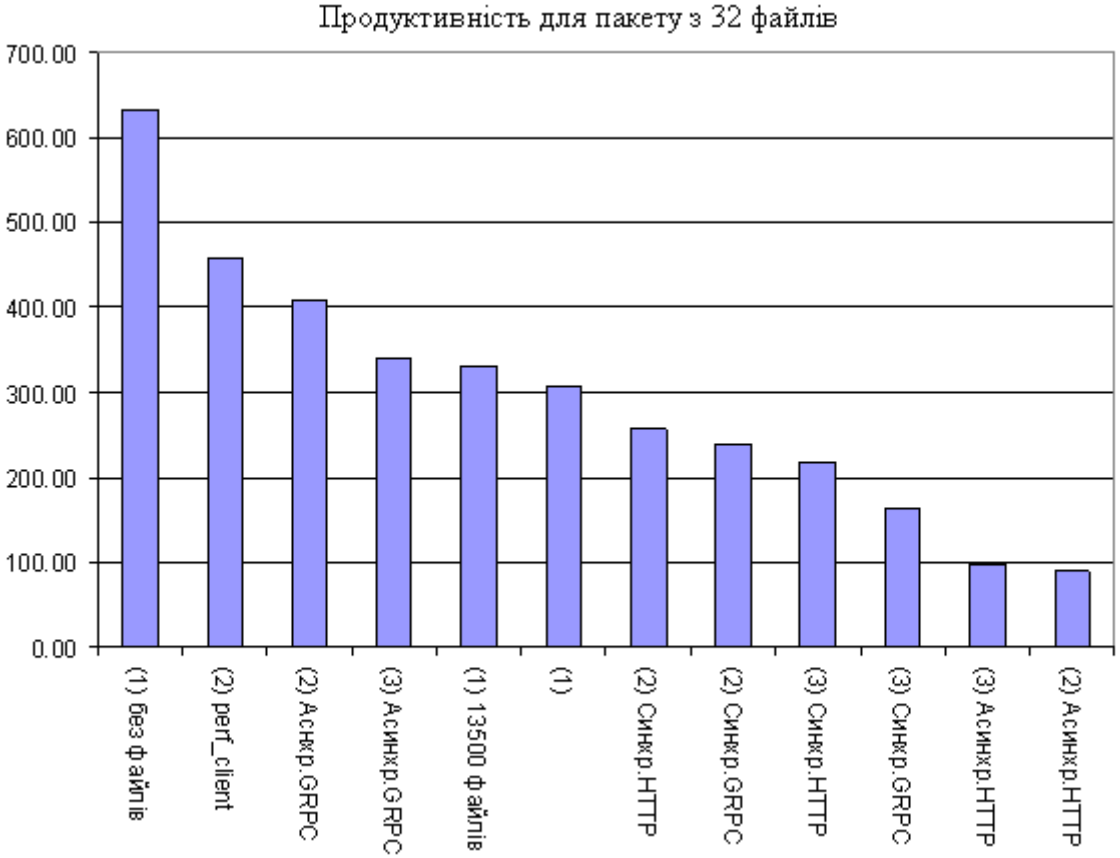

РИС. 5. Продуктивність ResNet-50 у різних конфігураціях

Якщо видалити затримку на завантаження файлів, простий режим працює надзвичайно швидко. Це стовпчик «(1) без файлів». Можна також побачити, що у синхронному режимі всі інші варіанти архітектур («(2) синхронно HTTP», «(3) синхронно HTTP») працюють гірше за просту «(1)». Для більшої колекції «(1) 13500 файлів» проста архітектура також працює швидше. Ймовірно так впливає краща буферизація на рівні файлової системи, що встигає пристосуватися до послідовного завантаження файлів з однієї папки. Цікавим є також порівняння асинхронних режимів між собою. Залежно від мережевого протоколу вони є майже найшвидшими, чи взагалі найгіршими. Тобто йдеться про взаємне налаштування GPRC з контейнером від NVIDIA. Нарешті, «(2) perf_client» – це особливий тестовий клієнт, що надається NVIDIA у комплекті з Inference Server Docker для вимірювання продуктивності. Він не працює у розподіленій архітектурі, тому є підстави вважати, що perf_client використовує якісь недокументовані можливості взаємодії. До того ж він не завантажує файли, а надсилає в Inference Server якісь внутрішні дані (подібно до «(1) без файлів»). Порівнюючи з ним, можна побачити, що асинхронний режим з GPRC працює надзвичайно ефективно: різниця досить мала (458/410 кадрів на секунду для того ж вузла).

Крім власно продуктивності при дослідженнях вимірювалось і оцінювалось багато інших параметрів (табл. 2), переважно з метою виявити і ліквідувати вузькі місця.

ТАБЛИЦЯ 2. Додаткові параметри, що вимірювалися в експерименті

Вимірювання параметрів		**Архітектура**		
де	**що**	**(1)**	**(2)**	**(3)**
Прикладна програма	Час підготовки моделі	+	+	+
	Час завантаження і підготовки даних	+	+	+
Прикладна програма (залежить від сервера)	Час надсилання даних на розпізнавач		+	+
	Час завантаження результатів з розпізнавача		+	+
	Час роботи розпізнавача	+	+	+
Прикладна програма	Час друку протоколу		+	+
Разом на сервері	Середнє використання CPU		+	+
	Максимальне використання CPU		+	+
	Мінімальне використання CPU		+	+
Розпізнавач	Середнє використання CPU	+	+	+
	Максимальне використання CPU	+	+	+
	Мінімальне використання CPU	+	+	+
Разом на клієнті	Середнє використання CPU			+
	Максимальне використання CPU			+
	Мінімальне використання CPU			+
Прикладна програма	Середнє використання CPU	+	+	+
	Максимальне використання CPU	+	+	+
	Мінімальне використання CPU	+	+	+
GPU (на сервері)	Середнє використання GPU	+	+	+
	Максимальне використання GPU	+	+	+
	Мінімальне використання GPU	+	+	+
Разом на сервері або клієнті (без докера)	Середнє використання пам'яті	+	+	+
	Максимальне використання пам'яті	+	+	+
	Мінімальне використання пам'яті	+	+	+
Разом у докері	Середнє використання пам'яті		+	+
	Максимальне використання пам'яті		+	+
	Мінімальне використання пам'яті		+	+

Зокрема, було виявлено, що обсяг пам’яті сервера є критичним ресурсом для асинхронних режимів GPRC для обох архітектур з Inference Server Docker, особливо для коротких пакетів. Для довжини пакету 1 перевищення максимальної потреби у пам’яті щодо синхронного режиму сягало 4 – 4.4 разів. У точній відповідності до різниці продуктивності середнє завантаження GPU в простій архітектурі складало близько 50 %, а у інших архітектур при асинхронному режимі з GPRC – 75 – 80 %. А от звичайні процесори виявилися стабільно недовантажені. У всіх варіантах наявність лише 4 ядер не призвела б до помітних затримок.

Стабільність вимірювань оцінювалась як стандартна похибка з 20 однакових експериментів. Для простої і мікросервісної архітектур, тобто коли всі процеси відбувалися в одному вузлі, продуктивність виявилась стійкою, похибка не перевищувала 0,5 %. Натомість у мережі (розподілена архітектура) похибка зростала від 3,5 – 4,5 % для коротких пакетів до 20 – 35 % для великих

пакетів. Випадкова затримка передачі картинок у розподіленій архітектури може пояснити спад її продуктивності щодо мікросервісної для пакетів довших 10 кадрів (рис. 4).

Висновки. Результати оптимізації параметрів різних архітектур для машинного навчання дозволили якісно спроектувати СКІТ 4.5 AI – сегмент суперкомп'ютерного комплексу СКІТ [5] призначеного для задач штучного інтелекту. З його використанням вже отримані результати в кількох напрямках машинного навчання:

- досліджено масштабованість нейронних мереж з шарами типів LSTM, згортки, та повний перцептрон, визначено залежність ефективності розпаралелювання від розміру пакету даних у системах з багатьма GPU на одному вузлі;

- побудована нейромережна модель для оцінки параметрів пористого середовища за даними акустичних досліджень свердловин (за синтетичними даними помилка прогнозу – 1.7 %) [6];

- побудована модель машинного навчання для аналізу великих даних телекомунікаційної компанії [7];

- реалізована нейромережна модель розріджено-розподіленої пам'яті на сучасних графічних процесорах і досліджені її параметри [8].

Список літератури

1. База даних зображень ImageNet. http://www.image-net.org (accessed Jan. 01, 2020).
2. He K., Zhang X, Ren S., Sun J. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE. 2016. https://arxiv.org/abs/1512.03385
3. Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M.S., Berg A.C., Li F. Imagenet large scale visual recognition challenge. *Computing Research Repository (CoRR)*. Ithaca, NY, USA: Cornell University. 2014. https://arxiv.org/abs/1409.0575
4. Mikami H., Suganuma H., U-chupala P., Tanaka Y., Kageyama Y. Massively Distributed SGD: ImageNet/ResNet-50 Training in a Flash. *Machine Learning*. Ithaca, NY, USA: Cornell University. 2019. https://arxiv.org/abs/1811.05233
5. Головинський А.Л., Сергієнко І.В., Тульчинський В.Г., Маленко А.Л., Бандура О.Ю., Горенко С.О., Роганова О.Ю., Лаврікова О.І. Розвиток суперкомп'ютерів серії СКІТ, розроблених в Інституті кібернетики імені В.М. Глушкова НАН України з 2002 по 2017 рр. *Кібернетика і системний аналіз*. 2017. 4. С. 124 – 129. http://www.kibernetika.org/volumes/2017/numbers/04/articles/12/ArticleDetailsEU.html
6. Khalimendik V. Porosity structure prediction from conventional sonic well logs on the base of synthetic samples computed by Prodaivoda-Maslov's method. 18th International Conference on Geoinformatics – Theoretical and Applied Aspects (Kyiv, May 2019). EAGE. 2019. https://doi.org/10.3997/2214-4609.201902061
7. Лавренюк А.М., Лавренюк С.І. Оптимізація підбору параметрів моделей для аналізу великих даних телекомунікаційної компанії. XIII Міжнародна науково-технічна конференція *"Перспективи телекомунікацій"* (ПТ-2019). К.: КПІ ім. Ігоря Сікорського. 2019. С. 230 – 232. http://conferenc.its.kpi.ua/2019/paper/view/15736
8. Вдовиченко Р.О. Реалізація Розріджено-розподіленої пам'яті на сучасних графічних процесорах і дослідження характеристик моделі. *Комп'ютерна математика*. 2019. Вип. 1. С. 77 – 84. http://dspace.nbuv.gov.ua/handle/123456789/161936

Одержано 08.01.2020

Тульчинський Вадим Григорович,
доктор фізико-математичних наук, завідувач відділу
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,

Лавренюк Сергій Іванович,
кандидат фізико-математичних наук, старший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,

Роганов Вячеслав Юрійович,
кандидат фізико-математичних наук, науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,

Тульчинський Петро Григорович,
кандидат фізико-математичних наук, старший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,

Халімендік Валерій Валерійович,
молодший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ.
dep145@gmail.com

УДК 004.89

В.Г. Тульчинский, С.И. Лавренюк, В.Ю. Роганов, П.Г. Тульчинский, В.В. Халимендик

ФАКТОРЫ ПРОДУКТИВНОСТИ ПРИМЕНЕНИЯ МОДЕЛЕЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ОБЛАКЕ С ИСПОЛЬЗОВАНИЕМ GPU

Институт кибернетики имени В.М. Глушкова, Киев, Украина
Переписка: dep145@gmail.com

Введение. В работах по машинному обучению (МН) и искусственному интеллекту (ИИ) ударение обычно ставится на качестве классификации, или точности оценки параметров. Если внимание уделяется производительности, то преимущественно также говорится о производительности этапа тренировки модели. Однако с распространением приложений ИИ в реальных прикладных задачах важнее становится проблема обеспечения высокой производительности обработки данных с помощью готовых натренированных моделей. По своему характеру эта проблема принципиально отличается от проблемы тренировки моделей: последняя имеет дело с интенсивными вычислениями, а первая – с простыми вычислениями, но большими потоками данных (файлов), поступающих из сети, или файловой системы на обработку. То есть это – типичная задача параллельной обработки с интенсивным вводом-выводом. Добавим, что с точки зрения прикладного применения, модуль ИИ, выполняющий классификацию, оценку, другую обработку данных является «черным ящиком»: стоимость разработки и тренировки модели, как и риски неудачи слишком высоки, чтобы заниматься такими задачами непрофессионально. Поэтому оптимизация производительности прежде всего предполагает подбор и балансировку параметров среды. Облачные системы с их гибкостью, управляемостью и легким масштабированием являются идеальными площадками для таких задач. Рассмотрим подробнее задачу исследования факторов, влияющих на производительность на одном, но значительном примере классификации выборки из коллекции изображений ImageNet [1] с помощью нейронной сети глубокого обучения по остаткам с 50 слоями – ResNet-50 [2].

Цель работы. Экспериментально исследовать факторы, влияющие на производительность применения готовых нейросетевых моделей в облачных системах различной архитектуры с графическими ускорителями.

Результаты. Оценены накладные расходы, связанные с микросервисной и распределенной архитектурами, влияние памяти, сети, размера пакетов, синхронного и асинхронного взаимодействия. Продемонстрирован сложный нелинейный характер влияния параметров системы в различных комбинациях.

Ключевые слова: машинное обучение, облачные технологии, графические ускорители, GPU, системная архитектура, производительность.

UDC 004.89

V. Tulchinsky, S. Lavreniuk, V. Roganov, P. Tulchinsky, V. Khalimendik

FACTORS OF PERFORMANCE FOR APPLICATION OF AI MODELS IN GPU CLOUD

V.M. Glushkov Institute of Cybernetics, Kyiv, Ukraine
Correspondence: dep145@gmail.com

Introduction. In machine learning (ML) and artificial intelligence (AI) works, the emphasis is usually on the quality of classification or the accuracy of parameter estimation. If the focus is on performance, then it is also mainly about the performance of the model's training phase. However, with the proliferation of AI applications in real-world problems, the problem of ensuring high data processing performance with ready models becomes more important. By its nature, this problem is fundamentally different from the one of model training: the latter deals with intensive calculations and the former with simple calculations, but large flows of data (files) coming from the network or file system for processing. That is, the typical task of parallel processing with intensive input-output. Besides, in terms of application, the AI module that performs classification, evaluation, or other data processing is a "black box": the cost of developing and training the model, as well as the risks of failure, are too high to handle such tasks in a non-professional manner. Therefore, performance optimization primarily involves the selection and balancing of system parameters. Cloud systems with their flexibility, manageability and easy scaling are the ideal platforms for such tasks. Consider in more detail the task of investigating the factors which affect performance on a single, but notable, pattern recognition sample of a subset of ImageNet image collection [1] classified by the 50-layer deep learning neural network ResNet-50 [2].

The purpose of the paper is to experimentally investigate the factors that influence the performance of a ready-to-use neural network model application in GPU cloud systems of various architectures.

Results. Overheads related to microservices and distributed architectures, memory, network, batch size, synchronous and asynchronous interactions are estimated. The complex nonlinear nature of the influence of the system parameters in various combinations is demonstrated.

Keywords: machine learning, cloud technologies, GPU, system architecture, performance.