ISSN 2522-9818 (print)
ISSN 2524-2296 (online) *Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)*

УДК 004.75 DOI: https://doi.org/10.30837/ITSSI.2024.27.128

С. Мінухін, Н. Коптілов

МЕТОД ЗБІЛЬШЕННЯ ПРОДУКТИВНОСТІ *APACHE SPARK* НА ОСНОВІ СЕГМЕНТУВАННЯ ДАНИХ І НАЛАШТУВАНЬ КОНФІГУРАЦІЙНИХ ПАРАМЕТРІВ

У використанні сучасних інструментів оброблення великих даних виникає проблема підвищення продуктивності сучасних фреймворків у контексті ефективного налаштування різних конфігураційних параметрів. **Об'єктом дослідження** є обчислювальні процеси оброблення великих даних із застосуванням технологій надпродуктивних фреймворків. **Предметом** є методи та підходи до ефективного налаштування конфігураційних параметрів фреймворків в умовах обмежень середовищ віртуалізації та локального ресурсу. **Мета дослідження** полягає в підвищенні продуктивності режимів розгортання *Apache Spark* та *Apache Hadoop* на основі комбінованого підходу, що містить передпроцесне сегментування вхідних даних та налаштування основних та додаткових конфігураційних параметрів з огляду на обмеження віртуального середовища та локального ресурсу. Досягнення поставленої мети передбачає виконання низки **завдань**: 1) створити синтезований набір тестових даних *WordCount* для використання методів сегментування вхідної інформації; 2) визначити склад загальних та специфічних конфігураційних параметрів *Apache Spark* та *Apache Hadoop*, що найбільше впливають на продуктивність роботи фреймворків у режимах розгортання *Spark Standalone* та *Hadoop Yarn (FIFO)*; 3) обгрунтувати зміни значень конфігураційних параметрів (прийняті за замовчуванням) за допомогою налаштувань рівня паралелізму, кількості розбиттів вхідного файлу відповідно до кількості ядер процесора, кількості завдань, що призначаються на кожне ядро та виконавця в системі; 4) перевірити теоретичні результати та довести їх використання на практиці. У дослідженні впроваджено такі **методи**: статистичний аналіз; метод генерації тестових даних за визначеними характеристиками сегментування з довільними обсягами інформації; системний підхід для комплексного оцінювання та аналізу продуктивності фреймворків на основі обраних конфігураційних параметрів. **Результати**. На основі запропонованого методу вибору складу параметрів для оцінювання продуктивності досліджуваних фреймворків проведено експерименти, що передбачали: застосування методу сегментування вхідної інформації на основі розділення вхідного файлу на абзаци (рядки) для різних значень діапазонів кількості слів та кількості літер у кожному слові; налаштування основних параметрів та специфічних, зокрема партиціонування та паралелізму з огляду на характеристики віртуального середовища та локального ресурсу. За досягнутими результатами детально проаналізовано запропоновані методи, впроваджені для покращення продуктивності досліджуваних фреймворків із рекомендаціями вибору оптимальних значень параметрів сегментування даних та конфігураційних параметрів. **Висновки**. Упровадження запропонованих методів налаштування конфігураційних параметрів *Spark* та *Hadoop* дає змогу підвищити продуктивність оброблення даних: для невеликих файлів (0,5–1 ГБ) у середньому до 25–30%; для великих (1,5–2,5 ГБ) – у середньому до 10–20 %. Водночас середнє значення часу виконання одного завдання зменшилося на 10–15 % для файлів різних розмірів та з різною кількістю слів у рядку.

Ключові слова: фреймворк; вхідний файл; сегментування; тестові дані; генератор даних; час виконання; конфігураційні параметри; *Spark; Hadoop; MapReduce*.

Актуальність дослідження

З настанням ери великих даних технології їх оброблення в останні роки привертають все більше уваги з боку науковців, світового бізнесу тощо [1]. Наприклад, *Google* пропонує три програмні технології для масового зберігання мультимедійних даних: *Google File System (GFS), MapReduce* та *BigTable* [2]. Традиційна обчислювальна модель в епоху великих даних вже не відповідає вимогам до продуктивності та ефективності їх використання. Отже, *Apache Spark* [3], *Apache Hadoop* [4], *Apache Storm* [5] та інші розподілені фреймворки натепер набули значного поширення завдяки потужним механізмам щодо оброблення значних обсягів інформації. Серед цих фреймворків *Apache Spark* став найбільш популярною та універсальною платформою для оброблення великих даних завдяки своїй продуктивності та значній підтримці прикладних сценаріїв використання.

Щодо аналізу великих даних, то потреба в інфраструктурі, здатній обробляти значні обсяги інформації за прийнятний час і з обмеженими ресурсами, є значною проблемою. Можливі рішення передбачають використання технологій паралельних

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27) ISSN 2524-2296 (online)

і розподілених обчислень. Прикладом застосування таких технологій є *Hadoop* – екосистема для реалізації паралельного програмування та розподіленого зберігання інформації. Однією з основних переваг використання *Hadoop* є те, що можна об'єднати процеси зберігання та оброблення інформації [6].

Одним із сучасних світових напрямів щодо використання високопродуктивних систем є застосування віртуальних середовищ для оброблення великих даних, що позначилося на розвитку технологій віртуалізації, упроваджених на хмарних платформах. Поряд із цим світовим трендом набули значного розвитку методи та технології використання віртуалізації в планувальниках для оброблення значних обсягів інформації [7, 8]. Хоча існує певний потенціал низької продуктивності та високого навантаження, віртуальне середовище може бути застосовано для моделювання роботи фреймворків та розроблення напрямів з метою збільшення рівня використання наявних системних ресурсів, полегшення управління системою, а також підвищення надійності та оптимізації енергозбереження.

Отже, постає необхідність **загального розв'язання** зазначених проблем і пов'язаної з ними низки часткових завдань щодо використання сучасних фреймворків оброблення та зберігання великих даних та підвищення їх продуктивності в умовах застосування віртуальних середовищ на обмежених обчислювальних потужностях локальних ресурсів.

Стислий огляд і аналіз фреймворків оброблення великих даних (*Big Data*). Аналіз публікацій та визначення завдань

Аналіз стану проблеми буде проведено на основі фреймворків *Apache Spark* та *Hadoop*.

Apache Spark

Apache Spark має понад 150 конфігураційних параметрів, які можуть бути налаштовані користувачами відповідно до їхніх власних застосунків, щоб оптимізувати продуктивність фреймворку [3]. З одного боку, великий простір параметрів надає чимало можливостей для підвищення продуктивності завдяки ретельному налаштуванню конфігураційних параметрів, особливо щодо визначення складу найбільш впливових параметрів, а з іншого – приводить до необхідності визначення та врахування досить складних взаємодій (кореляцій, залежностей) між цими параметрами. Через необґрунтоване (або ж не доведене експериментальним чином конфігурації, зокрема за допомогою евристики) налаштування параметрів продуктивність *Spark*-застосунків доволі складно забезпечити відповідно до теоретичних значень параметрів, прийнятих за замовчуванням. Тому розроблення способів оптимізації продуктивності *Spark*-застосунків є актуальною проблемою, що заслуговує на ґрунтовні дослідження.

Apache Hadoop

Apache Hadoop – це програмний фреймворк із відкритим вихідним кодом, що реалізує обчислювальну модель *MapReduce*, яка підтримує надійні та масштабовані обчислення великих даних [9]. Фреймворк написаний мовою *Java* і містить такі основних модулі: *Hadoop Common, Hadoop Distributed File System*, *Yarn* та *MapReduce* [10, 11]. *Yarn* – звичайна обчислювальна тканина для підтримки *MapReduce* та застосунків в межах кластера *Hadoop. Yarn* дозволяє декільком застосункам працювати одночасно на спільному кластері й дає змогу їм узгоджувати ресурси за необхідністю [12]. Політики планування, що використовуються в системі *Hadoop YARN,* містять *FIFO, Fair* та *Capacity*. За політикою *FIFO* всі завдання, що очікують на виконання, сортуються в неспадному порядку за часом їх надходження. Усі запити на завдання від кожної вхідної роботи будуть впорядковані за їх пріоритетами, а також за розташуванням. При цьому, планувальники завдань по-різному впливають на продуктивність обчислень. Планувальник *FIFO* потребує більше часу для оброблення завдання порівняно з планувальниками *Fair* і *Capacity*, тоді як у планувальника *Fair* пропускна спроможність і час оброблення майже однаковий. Планувальник *Capacity* потребує більше часу виконання порівняно з планувальником *Fair*, тоді як планувальник *Fair* витрачає менше часу на завершення завдання порівняно з *FIFO*-планувальником. Оскільки планувальник *Capacity* використовує пропускну спроможність черги завдань, може статися так, що черги з меншою пропускною спроможністю вимагають більшого часу на виконання завдання. Планувальник *Fair* обробляє більше даних за секунду порівняно з планувальниками *Capacity* та *FIFO*. Емпіричне дослідження за такими планувальниками наведено в роботі [13]. У планувальнику *FIFO* завдання подаються в одну чергу й виконуються послідовно. Для призначення завдань планувальник *FIFO* дотримується чіткого порядку виконання завдань. Недоліком цього

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

планувальника є те, що "жорсткий" порядок завдань *FIFO* зменшує локальність даних, і наступні завдання у черзі призначаються вузлам тільки після завершення попереднього завдання. У роботі [14] реалізовано дві політики *Fair* планування в *Hadoop YARN*: справедлива (*Fair*) та домінантна (*DRF*). *Fair* політика бере до уваги лише використання пам'яті кожним завданням і намагається призначити рівну пам'ять, тоді як політика *DRF* спрямована на те, щоб гарантувати, що всі завдання отримають у середньому рівну частку домінантних вимог до ресурсів (наприклад, пам'яті чи пам'яті або ядер процесорів). Політика пропускної спроможності працює подібно до політики справедливості. За цією політикою планувальник намагається зарезервувати гарантовану потужність для кожного завдання і замовляє ці завдання відповідно до їх дефіциту (тобто розриву між очікуваною та фактичною пропускною спроможністю). У статті [14] автори запропонували нову систему класифікації для планувальників завдань, розділивши їх на три окремі групи: планувальники завдань для зменшення кількості відсталих завдань, планувальники завдань для покращення локалізації даних та планувальники завдань для оптимізації використання ресурсів. У роботі [15] увагу зосереджено на алгоритмах планування завдань у середовищі *Hadoop Big Data*. Автори наголошують на важливості ефективного планування завдань у процесі оброблення великих обсягів даних в реальному часі, зважаючи на обмеження традиційних алгоритмів планування.

Hadoop MapReduce – парадигма паралельного програмування, що досліджується в роботах [16–18]. Модулі *MapReduce* налаштовуються за допомогою конфігурації параметрів, які в сукупності визначають та забезпечують продуктивність, що надається застосункам. У роботах [19, 20] наведено результати проведених експериментів щодо аналізу продуктивності етапів реалізації *MapReduce* та рекомендацій щодо покращення продуктивності технологій *MapReduce*.

Технології віртуалізації для кластерів *Apache Spark*

Вимоги до продуктивності застосування віртуальних середовищ у використанні фреймворків детально наведено та проаналізовано в працях [21–24].

Таким чином, виникає необхідність вирішення питань щодо вибору складу та налаштувань конфігураційних параметрів розглянутих фреймворків з врахуванням обмежень наявних обчислювальних ресурсів, зокрема, при розгортанні у віртуальних середовищах, що дозволить на основі отриманих результатів розробити рекомендації для підвищення їх продуктивності.

Метою статті є підвищення продуктивності режимів розгортання *Apache Spark* та *Apache Hadoop* на основі методу, що містить передпроцесного сегментування вхідних даних та налаштування основних і спеціальних (додаткових) конфігураційних параметрів, що беруть до уваги обмеження віртуального середовища та локального ресурсу.

Матеріали та методи. Створення синтезованого набору тестових даних *WordCount*

Було обрано один з універсальних наборів тестових даних *WordCount*, які застосовуються для тестування фреймворків значних обсягів інформації [7, 25, 26]. Сегментування вхідних даних за абзацами (рядками) здійснено за допомогою програми, розробленої мовою *Python,* на основі випадкових чисел із визначеною кількістю слів в абзаці та різною кількістю літер у кожному слові. Це дало змогу згенерувати 100 000 варіантів сегментованого тексту за мінімальні проміжки часу генерації та відповідно до вимог до розмірів файлів тестових даних.

У табл. 1 наведено розміри тестового файлу, кількість слів в абзаці (рядку) тексту, кількість завдань (*tasks*), розрахованих для кожного тестового файлу відповідно до розміру блоку в системі *HDFS* (128 МБ) [4, 9]. Характеристики обчислювального середовища локального ресурсу для проведення експериментів подано в табл. 2. Програмне забезпечення віртуальної машини на локальному ресурсі: *Oracle VM VirtualBox v.7.0; ОС Ubuntu Xenial 16.04 LTS*, ОП – 6 ГБ; віртуальних ядер – 4.

Таблиця 1. *Кількість завдань у вхідних файлах*

Розмір файлу, ГБ	Кількість завдань	Кількість слів в абзаці (рядку), діапазони
0,5	4	100–150; 150–200; 200–300
1	8	100–150; 150–200; 200–300
1,5	12	100–150; 150–200; 200–300
2	16	100–150; 150–200; 200–300
2,5	20	100–150; 150–200; 200–300

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Таблиця 2. *Характеристики обчислювального середовища локального ресурсу*

CPU Type	Intel Core i5 8265U
#Cores	4 cores@ 1.60GHz
#Threads	8 threads
#Sockets	Socket 1356 FCBGA
L1 DCache	4 x 32KB, 8-way associative, 64 byte/line
L1 ICache	4 x 32KB, 8-way associative, 64 byte/line
L2 Cache	4x 256 KB, 4-way associative, 64 byte/line
L3 Cache	L3 Cache 6 MB, 12-way associative, 64 byte/line
Memory	8 GB DDR4
Network	100 MB Ethernet link

Склад конфігураційних параметрів та їх налаштувань для *Spark* і *Hadoop Map Reduce*. Моделювання режимів *Spark Standalone* та *Hadoop Yarn FIFO*

На основі проведеного аналізу щодо визначення найбільш впливових параметрів налаштувань досліджуваних фреймворків у розглянутих режимах [27] обрано склад, наведений у табл. 3.

Для подальшого дослідження проведено статистичний аналіз результатів моделювання роботи режимів *Spark Standalone* та *Hadoop Yarn FIFO* для різних розмірів вхідних файлів та кількості слів у абзаці (рядку) сегментованого тексту (табл. 4–6) за репрезентативною кількістю проведених випробувань, яка дорівнювала 10 експериментам. Для порівняльного аналізу використано такі статистичні показники: МО часу виконання ±С.К.В за кожним режимом та відношення значень цих показників для різних режимів з метою визначення зростання продуктивності (часу виконання завдань) за допомогою розрахунку значення МО±С.К.В. як відношення МО_СКВ *Standalone* до МО_СКВ *Yarn*. Для цього застосовано такі визначення: МО – математичне очікування; С.К.В. – середньоквадратичне відхилення. Наведені обчислення є результатами застосування запропонованих налаштувань згідно з табл. 3.

Таблиця 3. *Найбільш впливові параметри налаштувань досліджуваних фреймворків*

Hadoop			***Spark***		
Параметр	**Значення за замовчуванням**	**Налаштоване значення**	**Параметр**	**Значення за замовчуванням**	**Налаштоване значення**
dfs.block.size	128 МБ	128 МБ	dfs.block.size	128 МБ	128 МБ
dfs.replication	3	1	dfs.replication	3	1
mapreduce.reduce.java.opts	1024	1638	SPARK_DRIVER_MEMORY	512 МБ	1 ГБ
mapreduce.map.cpu.vcores	2	4	SPARK_WORKER_INSTANCES	1	2
mapreduce.reduce.cpu.vcores	1	4	SPARK_WORKER_CORES	All available	4
mapreduce.reduce.shuffle.parallelcopies	5	5	SPARK_EXECUTOR_INSTANCES	2	2
mapreduce.task.io.sort.mb	100 МБ	100 МБ	SPARK_EXECUTOR_MEMORY	1 ГБ	1 ГБ
yarn.scheduler.maximum-allocation-mb	8192 МБ	4096 МБ	SPARK_EXECUTOR_CORES	1	1
sort. spill. percent	80%	80%	spark.shuffle.compress	true	true
mapreduce.map.memory.mb	1024	4096	spark.shuffle.spill.compress	true	true
mapreduce.reduce.memory.mb	1024	4096	spark.shuffle.file.buffer	32 КБ	48 КБ
yarn.app.mapreduce.am.resource.mb	1536	4096	spark.memory.fraction	0.6	0.4
yarn.nodemanager.resource.memory-mb	8192	4096	spark.memory.storageFraction	0.5	0.6
			spark.rdd.compress	true	true
			spark.io.compression.code	lz4	lz4
			spark.default.parallelism	4	4–100
			spark.sql.shuffle.partitions	4	4
			spark.task.cpus	1	2

ISSN 2522-9818 (print)
ISSN 2524-2296 (online)
Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

Таблиця 4. *Результати розрахунків для кількості слів у рядку* ***150–200***

Розмір, ГБ	Кількість завдань	МО_СКВ *Standalone*	Середній час виконання 1 завдання, с, *Standalone*	МО_СКВ *Yarn*	Середній час виконання 1 завдання, с, *Yarn*	МО ±С.К.В. відношення
0.5	4	**88.549**±0.936	**22**	**72.588**±1.292	**18**	**1.25**±0.025
1	8	**126.559**±1.336	**15,75**	**115.7374**±0.525	**14,75**	**1.092**±0.013
1.5	12	**168.593**±1.252	**14**	**151.026**±2.72	**12,6**	**1.1237**±0.022
2	16	**221.695**±1.399	**13,8**	**203.078**±0.826	**12,7**	**1.09163**±0.082
2.5	20	**265.94**±0.829	**13,25**	**240.382**±1.308	**12**	**1.10895**±0.0069

Таблиця 5. *Результати розрахунків для кількості слів у рядку* ***100–150***

Розмір, ГБ	Кількість завдань	МО_СКВ *Standalone*	Середній час виконання 1 таска, с, *Standalone*	МО_СКВ *Yarn*	Середній час виконання 1 таска, с, *Yarn*	МО ±С.К.В. відношення
0.5	4	**87.02**±0.632	**21, 75**	**67.803**±0.920	**16,75**	**1.303**±0.02
1	8	**132.54**±0.652	**16,5**	**119.569**±0.594	**14,9**	**1.1025**±0.0077
1.5	12	**194.44±**0.624	**16,2**	**172.915**±0.916	**14,33**	**1.1268**±0.007
2	16	**230.19**±0.632	**14,38**	**210.177**±0.852	**13,12**	**1.097**±0.0054
2.5	20	**265.02**±0.631	**13,25**	**253.15**±0.632	**12,65**	**1.047**±0.0037

Таблиця 6. *Результати розрахунків для кількості слів у рядку* ***200–300***

Розмір, ГБ	Кількість завдань	МО_СКВ *Standalone*	Середній час виконання 1 завдання, с, *Standalone*	МО_СКВ *Yarn*	Середній час виконання 1 завдання, с, *Yarn*	МО ± С.К.В. відношення
0.5	4	**94.85**±0.678	**23,5**	**66.604**±1.871	**16,5**	**1.525**±0.0412
1	8	**158.334**±0.633	**19,75**	**123.035**±0.515	**15,4**	**1.289**±0.0074
1.5	12	**204.922**±0.951	**17**	**174.231**±0.468	**14,5**	**1.177**±0.0063
2	16	**327.479**±0.474	**20,43**	**212.509**±0.557	**13,25**	**1.542**±0.0046
2.5	20	**338.079**±0.748	**16,9**	**255.299**±0.595	**12,75**	**1.325±**0.00426

На рис. 1–3 наведено графіки залежності часу оброблення вхідних файлів для різних режимів розгортання *Spark Standalone* та *Hadoop Yarn* (за замовчуванням), на рис. 4–6 – з використанням запропонованих налаштувань конфігураційних параметрів для різної кількості слів в абзаці (рядку).

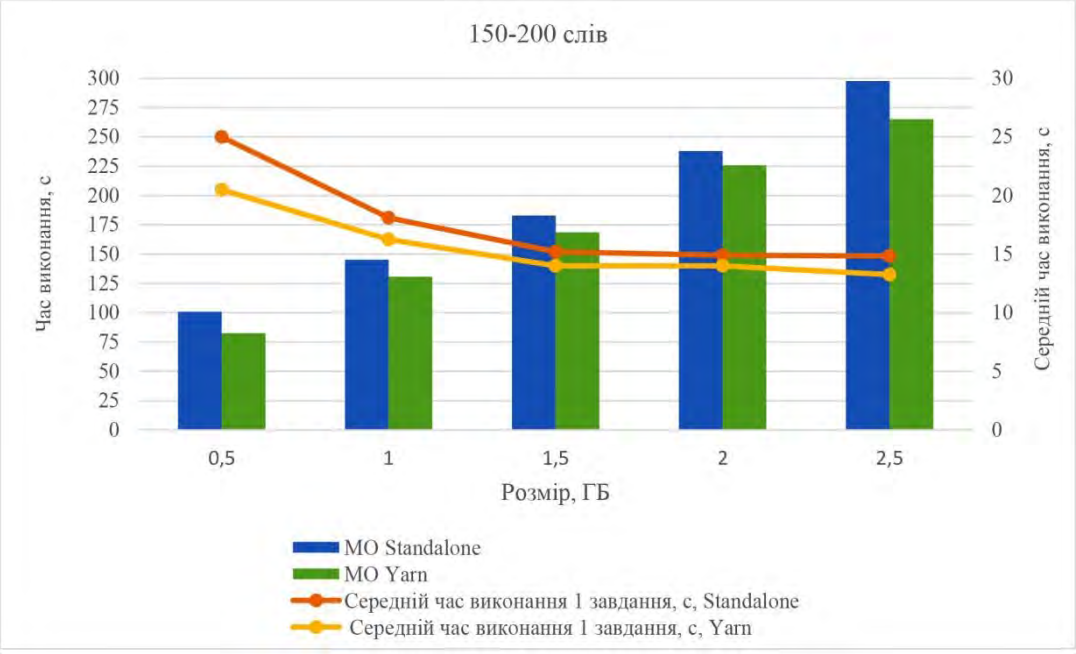

Рис. 1. Залежність часу оброблення вхідного файлу та середнього часу виконання завдання від розмірності вхідного файлу для різних режимів (150–200 слів)

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

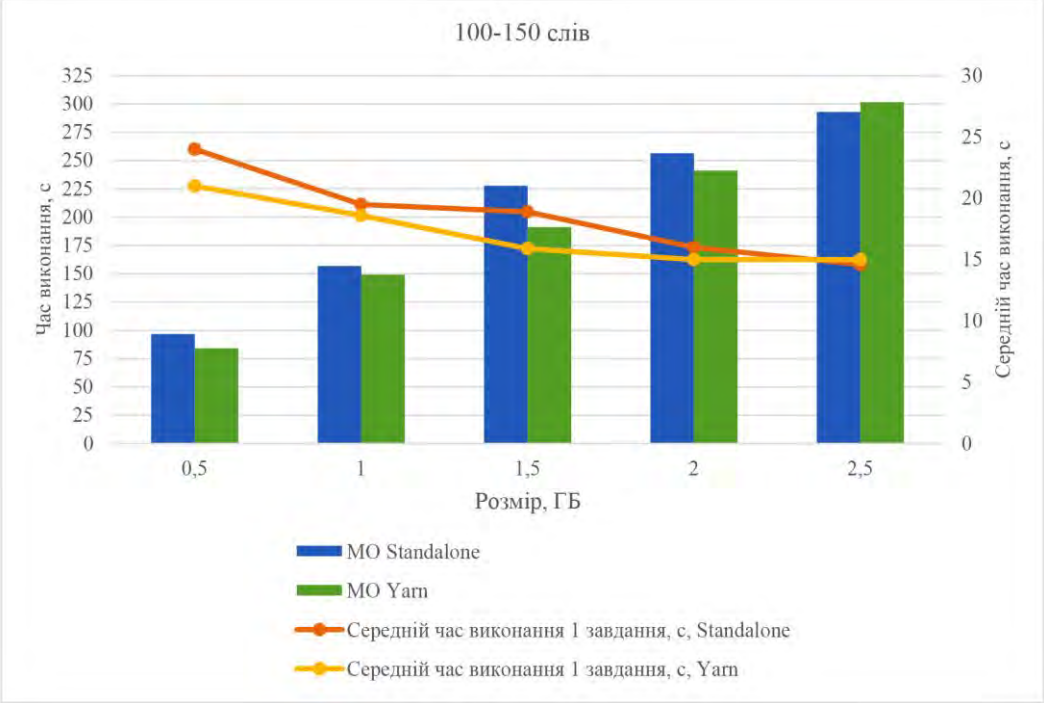

Рис. 2. Залежність часу оброблення вхідного файлу та середнього часу виконання завдання від розмірності вхідного файлу для різних режимів (100–150 слів)

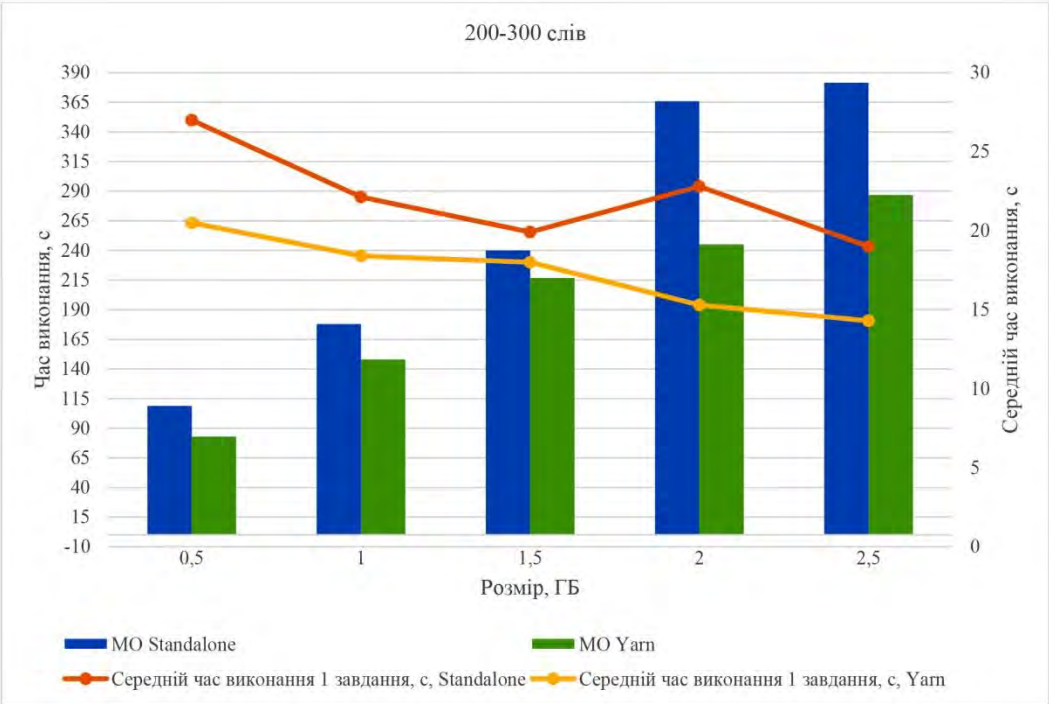

Рис. 3. Залежність часу оброблення вхідного файлу та середнього часу виконання завдання від розмірності вхідного файлу для різних режимів (200–300 слів)

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) *Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)*

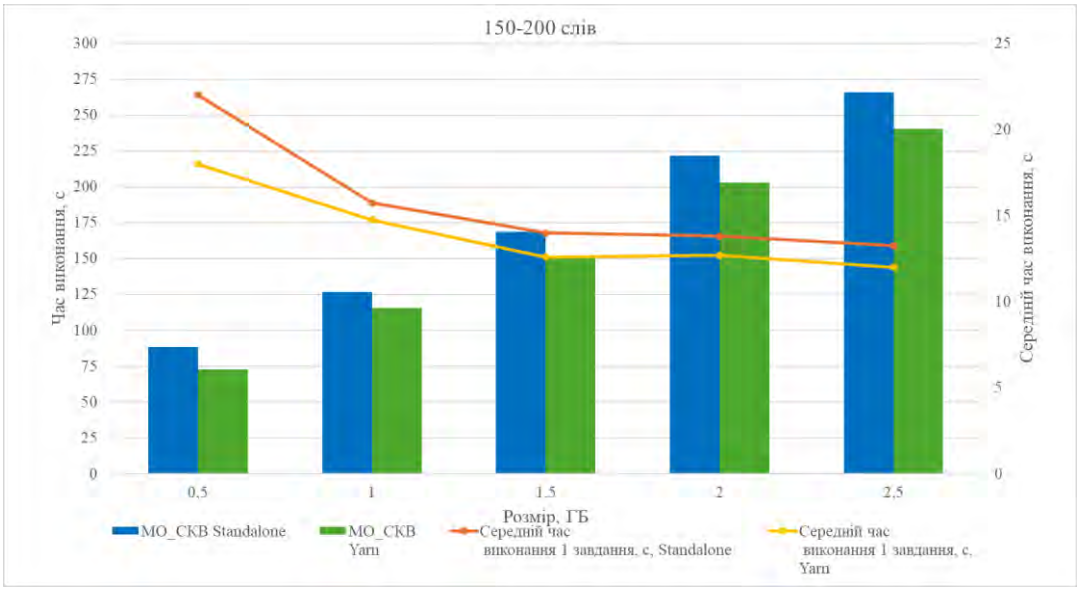

Рис. 4. Залежність часу оброблення вхідного файлу та середнього часу виконання завдання від розмірності вхідного файлу для різних режимів (150–200 слів)

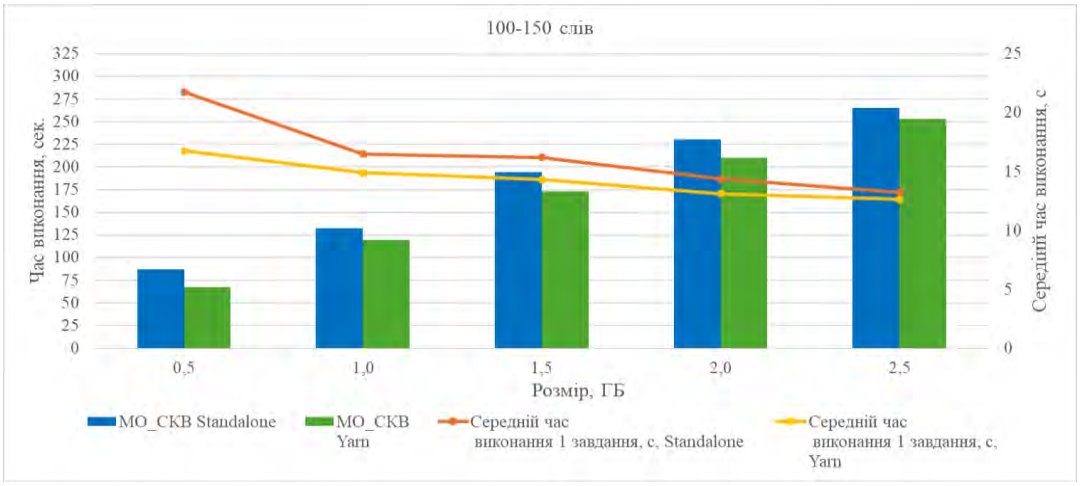

Рис. 5. Залежність часу оброблення вхідного файлу та середнього часу виконання завдання від розмірності вхідного файлу для різних режимів (100–150 слів)

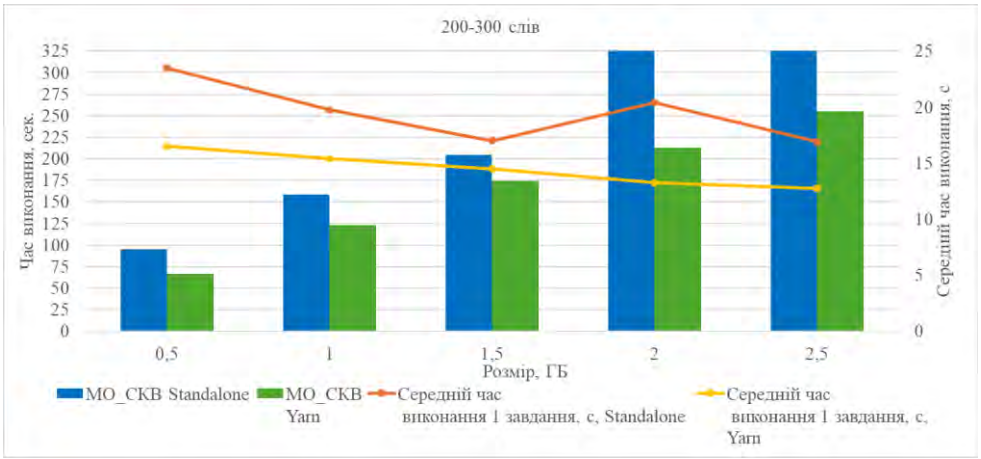

Рис. 6. Залежність часу оброблення вхідного файлу та середнього часу виконання завдання від розмірності вхідного файлу для різних режимів (200–300 слів)

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Порівняльний аналіз зроблено для налаштованих значень конфігураційних параметрів для режимів *Spark Standalone* та *Hadoop Yarn FIFO*.

Аналіз наведених результатів показує, що найбільш прийнятним щодо продуктивності є сегментування вхідного файлу в діапазоні 150–200 слів у рядку та застосування запропонованих налаштувань параметрів (див. табл. 3), що загалом сприяло підвищенню продуктивності оброблення: для невеликих файлів (0,5–1 ГБ) у середньому до 25–30%; для великих (1,5–2,5 ГБ) – у середньому до 10–20%. У цьому разі середнє значення часу виконання одного завдання зменшилося на 10–15% для файлів різних розмірів та з різною кількістю слів у рядку.

Аналіз впливу налаштувань конфігураційних параметрів розбиття та паралелізму

Унаслідок того, що в цьому дослідженні, на відміну від режиму *Single Node*, який передбачає використання одновузлового кластера, кластери *Spark Standalone* і *Hadoop Yarn FIFO* розгорнуті на трьох вузлах у віртуальному середовищі. Це визначає можливості використання додаткових конфігураційних параметрів – паралелізму і партиціонування, що підвищить продуктивність і масштабованість застосунків. Паралелізм дає змогу виконувати кілька завдань одночасно для одного вхідного файлу. Це означає, що замість послідовного оброблення інформації на одному вузлі можна розподілити завдання між трьома доступними вузлами з допомогою зміни *mapreduce.map.cpu.vcores (Hadoop)* і *spark.task.cpus (Spark)*, що прискорить час виконання. Партиціонування (*partitions*) розподіляє дані на логічні підмножини, що обробляються незалежно одна від одної. Це дає змогу розподілити навантаження між вузлами кластера та збільшити швидкість оброблення даних. *Spark* автоматично визначає кількість партицій, але для оптимізації продуктивності їх можна змінити вручну.

Для обґрунтування впливу налаштувань цих параметрів було змінено значення *Spark.task.cpus* та *Parallelism* у відповідних файлах налаштувань для режимів *Standalone* і *Hadoop Yarn FIFO* для різних розмірів вхідних файлів. Результати проведених експериментів для режиму *Standalone* наведено в табл. 7–9.

Як видно з наведених графіків, досягнуто **синергетичний ефект** запропонованих налаштувань завдяки динамічній зміні одночасно двох параметрів – *Spark.task.cpus* та *Parallelism*.

Таблиця 7. *Результати розрахунків часу для кількості слів у рядку* ***150–200***

Розмір файлу, ГБ	Spark.task.cpus	Parallelism=4	Spark.task.cpus	Parallelism=16
0,5	1	83.08	2	80.08
1	1	120.13	2	117.92
1,5	1	161.06	2	148.5
2	1	209.81	2	200.13
2,5	1	250.89	2	239.35

Таблиця 8. *Результати розрахунків часу для кількості слів у рядку* ***100–150***

Розмір файлу, ГБ	Spark.task.cpus	Parallelism=4	Spark.task.cpus	Parallelism=16
0,5	1	84.1	2	80.93
1	1	127.99	2	120.47
1,5	1	190.78	2	181.66
2	1	225.91	2	216.12
2,5	1	260.03	2	240.33

Таблиця 9. *Результати розрахунків часу для кількості слів у рядку* ***200–300***

Розмір файлу, ГБ	Spark.task.cpus	Parallelism=4	Spark.task.cpus	Parallelism=16
0,5	1	88.67	2	80.89
1	1	148.49	2	146.17
1,5	1	197.12	2	197.01
2	1	310.33	2	311.07
2,5	1	330.6	2	327.9

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

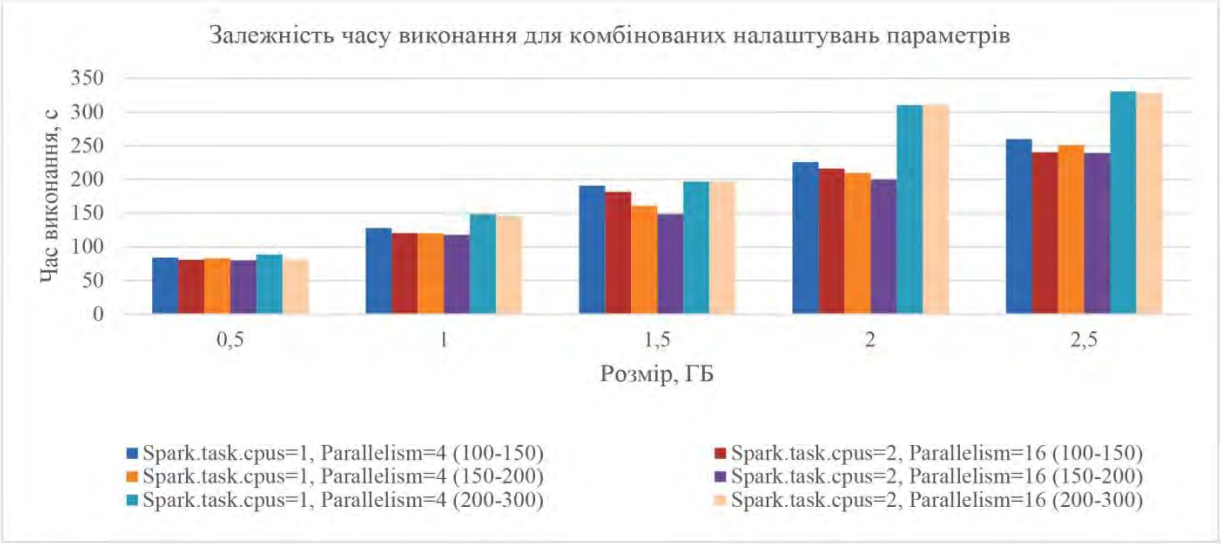

Рис. 7. Залежність часу виконання для комбінованих налаштувань параметрів для режиму *Standalone*

Висновки

З метою визначення та обґрунтування впливу масштабованості розмірів вхідної інформації створено синтезований набір тестових даних *WordCount* для використання методів сегментування вхідної інформації, що надходить для оброблення. Структура та характеристики тестів дають змогу використовувати результати оброблення інформації з динамічною структурою для збільшення продуктивності процесів оброблення великих даних. Це можна зробити з допомогою передпроцесної обробки даних на основі методів сегментування за абзацами (рядками, реченнями) за заданими діапазонами кількості рядків та слів у структурних елементах тексту.

Результати дослідження свідчать про ефективність запропонованих налаштувань основних та специфічних (додаткових) конфігураційних параметрів для кластерів *Spark Standalone* та *Hadoop Yarn FIFO*. У роботі обґрунтовано фактори, що впливають на актуальність окресленого питання. Вони полягають у використанні результатів моделювання роботи надпродуктивних кластерів в умовах обмежених обчислювальних ресурсів (віртуальних середовищ), налаштування яких потребує зважати на характеристики локального ресурсу (пам'ять, тип процесора, обсяг та організація кеш-пам'яті тощо). Це спричиняє обмеження щодо складу основних і додаткових конфігураційних параметрів, які впливають на загальну продуктивність системи. З допомогою експериментів обґрунтовано рекомендації щодо вибору діапазонів можливих змін значень параметрів в умовах обмеженості ресурсів, які визначаються спроможністю масштабувати ресурси та завдання. Запропоновано зміни додаткових параметрів налаштувань, що пов'язані з наявністю певних зв'язків між параметрами для підвищення продуктивності кластерів, зокрема рівнем розбиття даних, кількістю завдань для одного ядра, та відповідно до цього – рівня паралелізму. Результати, досягнуті для *Hadoop Yarn*, показали, що в окремих випадках збільшення кількості завдань для одного ядра не покращує продуктивність через обмеженість віртуальної машини або неможливість збільшити паралельне виконання потоків завдань (мініпотоків) на одному ядрі обчислювальної системи через архітектуру системи.

Практичне використання досягнутих результатів полягає в застосуванні напрямів та стратегій щодо вдосконалення методів вибору та налаштувань конфігураційних параметрів високопродуктивних фрейворків на основі детального аналізу зв'язків між параметрами з огляду на обмеження та умови їх розгортання та налаштування на наявних ресурсах.

Подальші напрями досліджень будуть спрямовані на побудову математичних моделей з метою визначення можливих варіантів комбінацій загальних і специфічних конфігураційних параметрів та їх налаштувань для створення сценаріїв підвищення продуктивності фреймворків в умовах масштабованості та з використанням різноманітних тестів для оброблення великих даних. Розроблений метод сприятиме розвитку методологічних засад для збільшення продуктивності сучасних технологій та інструментів оброблення великих даних.

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Список літератури

1. Borthakur D. Petabyte scale databases and storage systems at Facebook. *SIGMOD '13: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data.* 2013. P. 1267–1268. DOI: https://doi.org/10.1145/2463676.2463713
2. Survey A. Past, Present and Future of Hadoop / A. Zarei et al. *Computer Science. Networking and Internet Architecture*. 2022. DOI: https://doi.org/10.48550/arXiv.2202.13293
3. Apac he Spark Unified engine for large-scale data analytics. URL: http://spark.apache.org/
4. Apache Hadoop. URL: https://hadoop.apache.org/
5. Apache Storm. URL: https://storm.apache.org/2024/02/02/storm261-released.html
6. Polato I. A Comprehensive View of Hadoop Research – A Systematic Literature Review. / I. Polato et al. *Journal of Network and Computer Applications*. 2014. Vol. 46. P. 1–25. DOI: https://doi.org/10.1016/j.jnca.2014.07.022
7. Jeyaraj R., Ananthanarayana V. S., Paul A. Fine-grained data-locality aware MapReduce job scheduler in a virtualized environment. *Journal of Ambient Intelligence and Humanized Computing.* Vol. 11. 2020. P. 4261–4272. DOI: https://doi.org/10.1007/s12652-020-01707-7
8. Ibrahim S., Lu L., Qi L. Evaluating MapReduce on virtual machines: The Hadoop case. *IEEE International Conference on Cloud Computing*. Vol. 5931. 2009. P. 519–528. DOI: 10.1007/978-3-642-10665-1_47
9. White T. Hadoop: The definitive guide. O'Reilly Media, Inc. 2012. URL: https://www.academia.edu/34540716/Hadoop_The_Definitive_Guide
10. Vavilapalli V. K. Apache Hadoop YARN: Yet Another Resource Negotiator. / V. K. Vavilapalli et al. *SOCC '13: Proceedings of the 4th annual Symposium on Cloud Computing.* 2013. No 5. P. 1–16. DOI: 10.1145/2523616.2523633
11. Yi Yao New Scheduling Algorithms for Improving Performance and Resource Utilization in Hadoop YARN Clusters. / Yi Yao et al. *IEEE Transactions on Cloud Computing*. 2019. Vol. 9. №. 3. P. 1158–1171. DOI: 10.1109/TCC.2019.2894779
12. Perwej Y. An Empirical Exploration of the Yarn in Big Data. / Y. Perwej et al. *International Journal of Applied Information Systems (IJAIS)*. 2017. Vol. 12. № 9. P. 19–29. DOI: 10.5120/ijais2017451730
13. J. V. Gautam Empirical Study of Job Scheduling Algorithms in Hadoop MapReduce. / J. V. Gautam et al. *Cybernetics and Information Technologies*. 2017. Vol. 17. No 1. P. 146–163. DOI: 10.1515/cait-2017-0012
14. R. Ghazali A classification of Hadoop job schedulers based on performance optimization approaches. / R. Ghazali et al. *Cluster Computing.* 2021. Vol. 24, Issue 4. P. 3381–3403. DOI: https://doi.org/10.1007/s10586-021-03339-8
15. A. A. Abdallat Hadoop MapReduce Job Scheduling Algorithms Survey and Use Cases. / A. A. Abdallat et al. *Modern Applied Science*. 2019. Vol. 13. No. 7. P. 38–48. DOI: 10.5539/mas.v13n7p38
16. Abdul H. S. An overview on Big Data and Hadoop. *International Journal of Computer Applications.* Vol. 154. Number 10. 2016. P. 29–35. DOI: 10.5120/ijca2016912241
17. S. Hedayati MapReduce scheduling algorithms in Hadoop: a systematic study. / S. Hedayati et al. *Journal of Cloud Computing*. 2023. Vol. 12. Issue 1. P. 1–30. DOI: 10.1186/s13677-023-00520-9
18. M. Pastorelli Practical Size-based Scheduling for MapReduce Workloads. / M. Pastorelli et al. Computer Science. Distributed, Parallel, and Cluster Computing. 2013. 12 p. DOI: https://doi.org/10.48550/arXiv.1302.2749
19. Herodotou H., Babu S. Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs. *Proceedings of the VLDB Endowment.* Vol. 4. Issue 11. 2011. P. 1111–1122. DOI: 10.14778/3402707.3402746
20. H. Chang Scheduling in Mapreduce-Like Systems for fast completion time. / H. Chang et al. *NFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies.* DOI: https://doi.org/10.1109/INFCOM.2011.5935152
21. High-Performance Virtualized Spark Clusters on Kubernetes for Deep Learning. Performance Study. 2021. URL: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/spark-k8s-vsphere67-perf.pdf
22. Q. Zhang A Comparative Study of Containers and Virtual Machines in Big Data Environment. / Q. Zhang et al. 2018. URL: https://arxiv.org/pdf/1807.01842.pdf
23. S. A. Babu System performance evaluation of para virtualization, container virtualization, and full virtualization using Xen, Openvz, And Xenserver. / S. A. Babu et al. *In Advances in Computing and Communications (ICACC).* 2014. P. 247–250. DOI: 10.1109/ICACC.2014.66
24. J. Bhimani Accelerating big data applications using lightweight virtualization framework on enterprise cloud. / J. Bhimani et al. *In High Performance Extreme Computing Conference (HPEC).* 2017. P. 1–7. DOI: 10.1109/HPEC.2017.8091086
25. Issa J. Performance Evaluation and Estimation Model Using Regression Method for Hadoop Word Count. *IEEE Access.* Vol. 3. 2015. P. 2784–2793. DOI: 10.1109/ACCESS.2015.2509598
26. Benlachimi Y., Yazidi A. El, Hasnaoui M. L. A Comparative Analysis of Hadoop and Spark Frameworks using Word Count Algorithm. *International Journal of Advanced Computer Science and Applications.* Vol. 12. No. 4. 2021. P. 778–788. DOI: 10.14569/IJACSA.2021.0120495
27. Jayanthi M., Mohan R. K. R. Experimental Setup of Apache Spark Application Execution in a Standalone Cluster Environment using Default Scheduling Mode. *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS).* IEEE. 2022. P. 984–988. DOI: 10.1109/ICACRS55517.2022.10029155

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

References

1. Borthakur, D. (2013), "Petabyte scale databases and storage systems at Facebook", *SIGMOD '13: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data,* 2013. P. 1267–1268. DOI: https://doi.org/10.1145/2463676.2463713
2. Zarei, A., Safari,S., Ahmadi, M., Mardukhi F. (2022),"Past, Present and Future of Hadoop: A Survey". 2022. DOI: https://doi.org/10.48550/arXiv.2202.13293
3. Apache Spark. Unified engine for large-scale data analytics. available at: http://spark.apache.org/
4. Apache Hadoop. available at: https://hadoop.apache.org/
5. Apache Storm. available at: https://storm.apache.org/2024/02/02/storm261-released.html
6. Polato, I., Ré, R., Goldman, A., Kon F. (2014), "A Comprehensive View of Hadoop Research – A Systematic Literature Review", *Journal of Network and Computer Applications*. Vol. 46. P. 1–25. DOI: https://doi.org/10.1016/j.jnca.2014.07.022
7. Jeyaraj, R., Ananthanarayana, V. S., Paul, A. (2020), "Fine-grained data-locality aware MapReduce job scheduler in a virtualized environment", *Journal of Ambient Intelligence and Humanized Computing*. Vol. 11. P. 4261–4272. DOI: https://doi.org/10.1007/s12652-020-01707-7
8. Ibrahim, S., Lu, L., Qi, L. (2009), "Evaluating MapReduce on virtual machines: The Hadoop case", *IEEE International Conference on Cloud Computing*, 2009. Vol. 5931. P. 519–528. DOI: 10.1007/978-3-642-10665-1_47
9. White, T. "Hadoop: The definitive guide". O'Reilly Media, Inc. 2012. available at: https://www.academia.edu/34540716/Hadoop_The_Definitive_Guide
10. Vavilapalli, V. K., Murthyh, A. C., Douglasm, C., Agarwali, S., Konarh, M., Evansy, R., Gravesy, T., Lowey, J., Shahh, H., Sethh, S., Sahah, B., Curinom, C., O'Malleyh, O., Radiah, S., Reedf, B., Baldeschwielerh, E. (2013), "Apache Hadoop YARN: Yet Another Resource Negotiator", *SOCC '13: Proceedings of the 4th annual Symposium on Cloud Computing*, 2013. No 5. P. 1–16. DOI: 10.1145/2523616.2523633
11. Yao; Y., Gao, H.; Wang, J., Sheng, B., Mi, N. (2019), "New Scheduling Algorithms for Improving Performance and Resource Utilization in Hadoop YARN Clusters", *IEEE Transactions on Cloud Computing*. Vol. 9. №. 3. P. 1158–1171. DOI: 10.1109/TCC.2019.2894779
12. Perwej, Y., Kerim, B., Adrees, M. S., Sheta, O. E. (2017), "An Empirical Exploration of the Yarn in Big Data", *International Journal of Applied Information Systems (IJAIS).* Vol. 12. № 9. P. 19–29. DOI: 10.5120/ijais2017451730
13. Gautam, J. V., Prajapati, H. B., Dabhi, V. K., Chaudhary, S. (2017), "Empirical Study of Job Scheduling Algorithms in Hadoop MapReduce", *Cybernetics and Information Technologies*. Vol. 17. No 1. P. 146–163. DOI: 10.1515/cait-2017-0012
14. Ghazali, R., Adabi, S., Down, D. G., Movaghar, A. (2021), "A classification of Hadoop job schedulers based on performance optimization approaches", *Cluster Computing.* Vol. 24. Issue 4. P. 3381–3403. DOI: https://doi.org/10.1007/s10586-021-03339-8
15. Abdallat, A. A., Arwa, I. A., Duaa, A. A., amimi, AlWidian, J. A. (2019), "Hadoop MapReduce Job Scheduling Algorithms Survey and Use Cases", *Modern Applied Science*. Vol. 13. No. 7. P. 38–48. DOI: 10.5539/mas.v13n7p38
16. Abdul, H. S. (2016), "An overview on Big Data and Hadoop", *International Journal of Computer Applications*. Vol. 154. Number 10. P. 29–35. DOI:10.5120/ijca2016912241
17. Hedayati, S., Maleki, N., Olsson, T., Ahlgren, F., Seyednezhad, M., Berahmand, K. (2023), "MapReduce scheduling algorithms in Hadoop: a systematic study", *Journal of Cloud Computing*. Vol. 12. Issue 1. P. 1–30. DOI: 10.1186/s13677-023-00520-9
18. Pastorelli, M., Barbuzzi, A., Carra, D., Dell'Amico, M., Michiardi, P. (2013), "Practical Size-based Scheduling for MapReduce Workloads". 12 p. DOI: https://doi.org/10.48550/arXiv.1302.2749
19. Herodotou, H., Babu, S. (2011), "Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs", *Proceedings of the VLDB Endowment*. Vol. 4. Issue 11. P. 1111–1122. DOI: 10.14778/3402707.3402746
20. Chang, H., Kodialam, M., Kompella, R. R., Lakshman, T. V., Lee, M., Mukherjee, S., (2011), "Scheduling in Mapreduce-Like Systems for fast completion time", *NFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies,* 2011. DOI: https://doi.org/10.1109/INFCOM.2011.5935152
21. "High-Performance Virtualized Spark Clusters on Kubernetes for Deep Learning. Performance Study". 2021. available at: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/spark-k8s-vsphere67-perf.pdf
22. Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L., Zhou, W. "A Comparative Study of Containers and Virtual Machines in Big Data Environment". 2018. available at: https://arxiv.org/pdf/1807.01842.pdf
23. Babu, S. A., Hareesh, M. J., Martin, J. P., Cherian, S., Sastri, Y. (2014), "System performance evaluation of para virtualization, container virtualization, and full virtualization using Xen, Openvz, And Xenserver", *In Advances in Computing and Communications (ICACC),* 2014. P. 247–250. DOI: 10.1109/ICACC.2014.66
24. Bhimani, J., Yang, Z., Leeser, M., Mi, N. (2017), "Accelerating big data applications using lightweight virtualization framework on enterprise cloud", *In High Performance Extreme Computing Conference (HPEC)*. P. 1–7. DOI: 10.1109/HPEC.2017.8091086
25. Issa, J. (2015), "Performance Evaluation and Estimation Model Using Regression Method for Hadoop Word Count", *IEEE Access.* Vol. 3. P. 2784–2793. DOI: 10.1109/ACCESS.2015.2509598
26. Benlachimi, Y., Yazidi, A. El, Hasnaoui, M. L. (2021), "A Comparative Analysis of Hadoop and Spark Frameworks using Word Count Algorithm", *International Journal of Advanced Computer Science and Applications.* Vol. 12. No. 4. P. 778–788. DOI: 10.14569/IJACSA.2021.0120495

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

27. Jayanthi, M., Mohan, R. K. R. (2022), "Experimental Setup of Apache Spark Application Execution in a Standalone Cluster Environment using Default Scheduling Mode". *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*. P. 984–988**.** DOI: 10.1109/ICACRS55517.2022.10029155

Надійшла 10.03.2024

Відомості про авторів / About the Authors

Мінухін Сергій Володимирович – доктор технічних наук, професор, Харківський національний економічний університет ім. С. Кузнеця, професор кафедри інформаційних систем, Харків, Україна; e-mail: serhii.minukhin@hneu.net; ORCID ID: https://orcid.org/0000-0002-9314-3750

Коптілов Нікіта Сергійович – Харківський національний економічний університет ім. С. Кузнеця, магістрант, кафедра інформаційних систем, Харків, Україна; e-mail: koptilov.nikita.s@hneu.net; ORCID ID: https://orcid.org/0009-0009-2109-8717

Minukhin Serhii – Doctor of Sciences (Engineering), Professor, Simon Kuznets Kharkiv National University of Economics, Professor at the Department of Information Systems, Kharkiv, Ukraine.

Koptilov Nikita – Simon Kuznets Kharkiv National University of Economics, Master's Degree Student, Department of Information Systems, Kharkiv, Ukraine.

A METHOD TO ENHANCE APACHE SPARK PERFORMANCE BASED ON DATA SEGMENTATION AND CONFIGURATION PARAMETERS SETTINGS

When using modern big data processing tools, there is a problem of increasing the productivity of using modern frameworks in the context of effective setting of various configuration parameters. The object of the research is computational processes of processing big data with the use of technologies of high-performance frameworks. The subject is methods and approaches to the effective setting of configuration parameters of frameworks in the conditions of limitations of virtualization environments and local resources. The purpose of the study is to improve the performance of Apache Spark and Apache Hadoop deployment modes based on a combined approach that includes preprocess segmentation of input data and setting of basic and additional configuration parameters that take into account the limitations of the virtual environment and local resources. Achieving the set goal involves the following tasks: create a synthesized set of WordCount test data for using input data segmentation methods. Determine the composition of general and specific Apache Spark and Apache Hadoop configuration parameters that most affect the performance of frameworks in Spark Standalone and Hadoop Yarn (FIFO) deployment modes. Justify changes in the values of the configuration parameters (accepted by default) by setting the level of parallelism, the number of partitions of the input file according to the number of processor cores, the number of tasks assigned to each core and the system executor. Conduct experimental research to substantiate theoretical results and prove their use in practice. Methods. The research used the following methods: statistical analysis; a method of generating test data based on defined segmentation characteristics with arbitrary volumes of data; a systematic approach for comprehensive evaluation and analysis of performance of frameworks based on selected configuration parameters. The results. On the basis of the developed system of parameters for evaluating the performance of the studied frameworks, experiments were carried out, which include: the application of the method of segmentation of input data based on the division of the input file into paragraphs (lines) for different values of the ranges of the number of words and the number of letters in each word; setting the main parameters and specific ones, in particular, partitioning and parallelism, taking into account the characteristics of the virtual environment and the local resource. According to the obtained results, a detailed analysis of the use of the proposed methods to improve the performance of the studied frameworks with recommendations for choosing the optimal values of data segmentation parameters and configuration parameters was carried out. You are snowmen. The obtained results of the experiments allow us to conclude that the use of the proposed methods of setting the configuration parameters of Spark and Hadoop will increase the processing productivity: for small files (0.5–1 GB) on average up to 25–30%, for large ones (1.5–2.5 GB) – up to 10–20% on average. At the same time, the average value of the execution time of one task decreased by 10-15% for files of different sizes and with different number of words in a line.

Keywords: framework; input file; segmentation; test data; data generator; execution time; configuration parameters; Spark; Hadoop; MapReduce.

Бібліографічні описи / Bibliographic descriptions

Мінухін С. В., Коптілов Н. С. Метод збільшення продуктивності *Apache Spark* на основі сегментування даних і налаштувань конфігураційних параметрів. *Сучасний стан наукових досліджень та технологій в промисловості*. 2024. № 1 (27). С. 128–139. DOI: https://doi.org/10.30837/ITSSI.2024.27.128

Minukhin, S., Koptilov, N. (2024), "A method to enhance Apache Spark performance based on data segmentation and configuration parameters settings", *Innovative Technologies and Scientific Solutions for Industries*, No. 1 (27), P. 128–139. DOI: https://doi.org/10.30837/ITSSI.2024.27.128