СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ СИСТЕМ ПОТОКОВОГО АНАЛИЗА ДАННЫХ В ЗАДАЧЕ ОБРАБОТКИ ИЗОБРАЖЕНИЙ СКОЛЬЗЯЩИМ ОКНОМ

Казанский Н.Л., Проценко В.И., Серафимович П.Г.
Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет) (СГАУ),
Институт систем обработки изображений РАН

Аннотация

Проведён сравнительный анализ производительности и потребления памяти двух систем потоковой обработки данных: Apache Storm и IBM InfoSphere Streams в задаче обработки наборов изображений скользящим окном. Тестирование производилось на виртуальных машинах под управлением операционной системы CentOS. Выполнена оптимизация обеих систем потоковой обработки по потреблению памяти. Сделаны выводы об областях применимости рассмотренных систем.

Ключевые слова: большие данные, потоковая обработка, набор изображений, скользящее окно, используемая память.

Введение

На сегодняшний день интенсивность генерации данных различными компьютерными системами, например, на основе датчиков, видеокамер, гиперспектральной аппаратуры дистанционного зондирования Земли, может достигать нескольких десятков петабайт в секунду [1 – 3]. Традиционные подходы [4 – 6] к обработке больших данных, в частности, в соответствии с парадигмой MapReduce [7 – 9] не позволяют справиться с анализом потока такой интенсивности из-за ограничений на размер хранилищ данных и задержки на сохранение информации перед обработкой [10 – 14]. Системы потоковой обработки решают проблему, проводя анализ в реальном времени и обладая задержкой обработки элемента данных в несколько миллисекунд [15 – 17].

В последнее время появился ряд технологий, которые позволяют обрабатывать потоковые данные. Среди них можно назвать Storm, Akka, Finagle, MillWheel, Samza. Характерными представителями данного типа систем являются широко применяемые на практике бесплатный программный пакет Apache Storm и коммерческий пакет IBM InfoSphere Streams. К сожалению, количество опубликованных работ, содержащих анализ и сравнение систем, очень мало. Единственным широкодоступным источником сравнительного анализа на данный момент является бенчмарк «Of Streams and Storm» [16], в котором авторы сравнивают две системы в задаче обработки электронных писем как пример обработки потока текстовой информации. Не менее востребован и анализ других типов информации, например, изображений и видеопотоков [1 – 15]. Пакет Storm уже используется для обработки IP-видео компанией Comcast [17]. Компания IBM также включила в пакет InfoSphere Streams соответствующий модуль обработки изображений и видео.

Алгоритмы обработки изображений скользящим окном находят применение во многих задачах [18 – 20]. В настоящей статье проводится сравнение двух систем потоковой обработки Apache Storm и IBM InfoSphere Streams на простом алгоритме Гауссова размытия внутри окна, применяемого к набору изображений. Тестирование производится на виртуальных машинах с одноузловыми конфигурациями систем. Вначале запускаются реализации алгоритма в неоптимизированном виде, анализ которых приводит к оптимизированным реализациям. Производится сравнение по параметрам потребления памяти и пропускной способности. Это позволит сделать выводы об области применимости систем в зависимости от размера входных изображений и размеров доступной оперативной памяти.

Архитектуры Apache Storm и IBM InfoSphere Streams

Apache Storm

Архитектурно Storm состоит из двух *видов* управляющих процессов: локальных управляющих процессов Supervisor, по одному на каждый вычислительный узел, и центрального управляющего процесса Nimbus. На рис. 1 изображены основные компоненты системы. Задачами центрального управляющего процесса являются: назначение задач и рассылка исполняемого кода по машинам кластера, координация Supervisor процессов, мониторинг ошибок и распределение нагрузки в системе.

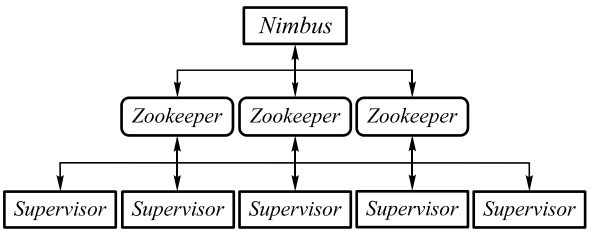

*Рис. 1. Архитектура **Storm**. Координация между главным и локальными управляющими процессами производится через сервис Zookeeper. Стрелками проиллюстрирована возможность обращения для записи и чтения к любому узлу Zookeeper*

Процессы Supervisor создают рабочие процессы для выполнения задач, отвечают за передачу и учёт кортежей между ними. Координация между Nimbus, Supervisor и рабочими процессами производится через Zookeeper [21] – сторонний сервис надёжного хранения конфигурационных данных. Кроме Zoo-

keeper, в системе для асинхронной передачи сообщений между рабочими процессами используется библиотека ZeroMQ [22] либо, начиная с версии 0.9.0, может быть использован Netty [23] – механизм передачи сообщений в качестве альтернативы. Передача сообщений (*кортежей*) между задачами внутри рабочих процессов осуществляется реализацией циклического буфера с активным ожиданием, именуемой LMAX Disruptor [24].

Потоковый алгоритм, называемый топологией (topology), представляет собой ориентированный граф, в узлах которого находятся объекты классов Spout (источник кортежей) или Bolt (обработчик кортежей), называемые *компонентами*. Разделение на источники и обработчики продиктовано необходимостью наличия механизма гарантированной обработки кортежей, для использования которого к каждому кортежу в источнике должен прикрепляться уникальный идентификатор, а во всех зависимых обработчиках после его обработки должен быть вызван метод подтверждения. Способ распределения кортежей между соединёнными узлами задаётся типом группировки (grouping) с равномерным распределением (shuffle) по умолчанию.

Фреймворк написан на языках Clojure и Java и исполняется на JVM. Благодаря хорошей совместимости с Java, разработчик для реализации алгоритма может свободно выбирать любой из этих языков. Источники данных и обработчики могут быть реализованы не на JVM-языках посредством протокола Thrift [25], но из-за дополнительных процедур сериализации и передач JSON-данных будут менее производительными по сравнению с JVM-аналогами.

IBM InfoSphere Streams

Программа представляет из себя граф, в узлах которого находятся операторы – основные единицы вычисления в системе InfoSphere Streams. Операторы могут быть реализованы на языках C++ и Java. Специальных средств для реализации на других языках фреймворк не предоставляет, такие реализации могут выполняться в системе, обёрнутыми в Java или C++. Результатом компиляции является набор раздёляемых библиотек (PE – Process Element), выполнение которых происходит в отдельных процессах и которые могут объединять несколько операторов после автоматической оптимизации (auto-fuse) либо после указания размещения операторов в одном разделе (partition).

На рис. 2 представлена схема архитектуры IBM InfoSphere Streams. В отличие от Storm управление на главном узле в архитектуре Streams производится не одним, а множеством сервисов, ориентированных на различные аспекты: сбор метрик производительности (SRM – Streams Resource Manager), назначение и завершение задач (SAM – Streams Application Manager), планирование выполнения задач в соответствии со статистикой производительности (SCH – Streams Scheduler), авторизация в системе и предоставление прав на операции ввода-вывода, запуск заданий (AAS – Authorization and Authentification Service), графический интерфейс с выводом метрик производительности (SWS – Streams Web Service). На выделенных для вычисления пользовательских алгоритмов узлах выполняется процесс PEC (Processing Element Container), являющийся контейнером для задач (PE). Контроль за назначением и выполнением задач, а также за сбором метрик производительности осуществляется в процессе HC (Host Controller). Recover DB – дополнительный сервис, который хранит состояние других сервисов в DB2 базе данных, что позволяет перезапустить их в случае аварийного завершения. Для создания пользовательской логики, устойчивой к ошибкам, может быть использован параметр конфигурации “checkpoint”, определяющий политику сохранения состояния оператора в общую файловую систему (shared file system). В качестве общей файловой системы могут быть выбраны Network File System (NFS) либо General Parallel File System (GPFS). Также общую файловую систему система использует для хранения своего состояния.

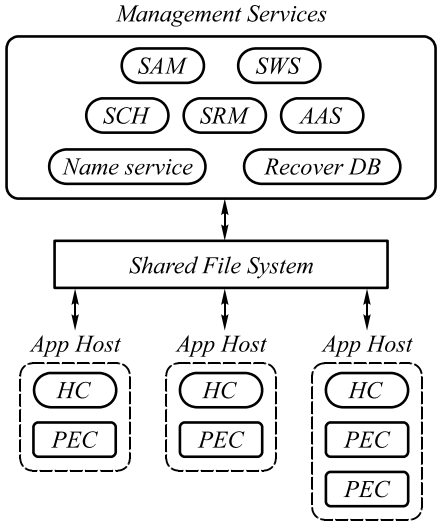

*Рис. 2. Архитектура **InfoSphere Streams**. Верхним прямоугольником изображается узел с управляющими сервисами, нижними пунктирными прямоугольниками изображены исполнительные узлы, каждый с локальным управляющим процессом (Host Controller) и рабочими процессами PEC. Взаимодействие производится через общую файловую систему (Shared File System)*

Вычислительный эксперимент

Описание

Целью данного вычислительного эксперимента являлся анализ производительности двух систем в задаче применения Гауссова фильтра к обработке потока изображений в рамках использования библиотеки OpenCV [26]. Сравнение производилось по двум параметрам: максимальное потребление памяти и пропускная способность.

Размер окна фильтра составлял 5×5 пикселей с параметрами дисперсии 3 по направлениям x и y.

Для системы Streams использовался размещённый в виде проекта с открытым кодом готовый набор операторов OpenCVToolkit [27], позволяющий реализо-

вать всю программу на языке SPL [28] без создания своих операторов для работы с библиотекой OpenCV. Версия кода для Storm написана на Clojure и использует обёртки над OpenCV динамическими библиотеками, созданные в процессе установки библиотеки.

Обе версии запускались для наборов из $N=1000$ квадратных изображений с размерами стороны в пикселях: 50, 100, 200, 400, 600, 800, 1000, 1200. Каждый набор содержал изображения одного размера.

Замер потребления памяти осуществлялся линукс-утилитой «top», а замер выделенного размера «кучи» JVM – утилитой «jvisualvm». Методика подсчёта пропускной способности «throughput» реализована операторами и одинакова в обеих версиях: при считывании и при записи в отдельный поток записывается кортеж с текущим временем $time_i$, где $i \in 1, 2, \ldots, 2N$, а затем по формуле

$$throughput = \frac{N}{\max\limits_i time_i - \min\limits_i time_i}$$

в дополнительных операторах рассчитывается искомое значение и записывается в файл. Единицей измерения пропускной способности было *количество изображений в секунду*. На рис. 3 изображена схема вычислений для систем Storm и Streams. На этапе «Read» чтение производилось с жёсткого диска, а на этапе «Write» изображения сохранялись в специальное устройство dev/null.png, запись в которое всегда успешна и не требует обращения к дискам.

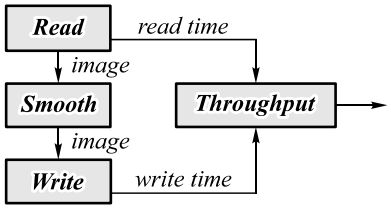

Рис. 3. Схема эксперимента

Топология Storm состояла из одного источника – «read-image» и трёх обработчиков: «smooth-image», «write-image», «throughput-bolt». Узлы «read-image», «smooth-image» и «write-image» образовывали конвейер по обработке изображений, а кортежи со временем чтения и записи изображения отправлялись через дополнительные потоки в «throughput-bolt».

Программа Streams состояла из последовательно соединённых операторов «DirectoryScan», «ReadImage», «Smooth», «SaveImage», отвечающих за обработку изображений. Пропускная способность рассчитывалась двумя операторами «Aggregate» и оператором «Barrier». Первые два подсчитывали максимальное время записи и минимальное время считывания в окне по времени из N кортежей. Затем оператор «Barrier» на основе полученных значений возвращал кортеж с искомой величиной, который сохранялся оператором «FileSink».

Тестирование проводилось на виртуальных машинах с операционной системой CentOS 6 с 4-ядерным процессором и 8 ГБ оперативной памяти. В этой связи приоритет отдавался анализу потребления памяти, как наименее подверженный изменению критерий при исполнении кода на реальном оборудовании. Хост-машина имела следующие параметры: 32 ГБ оперативной памяти и 6-ядерный процессор со включённым Hyper Threading. Использовалась библиотека OpenCV версии 2.4.9.

Также стоит отметить особенности настройки топологий. Для Storm каждый компонент выполнялся в своём потоке, значение параллелизма было равно 1. Количество JVM-процессов (TOPOLOGY_WORKERS) было равно 1, так что в эксперименте не использовались межпроцессовые механизмы передачи сообщений. Размер внутрипроцессовых очередей сообщений по умолчанию был равен 1024 элементами. В системе Streams операторы были объединены в 4 процесса (Process Element) – в первом процессе выполнялись «ReadDirectory» и «ReadImage», во втором – «Smooth», в третьем – «SaveImage» и в четвёртом – остальные, представляющие из себя логику подсчёта пропускной способности.

Результаты

В первую очередь, при создании бенчмарка были реализованы версии алгоритмов с максимальным количеством параметров по умолчанию, что совпадало со стремлением получить реализации с минимальными затратами на разработку. Запуск на тестовых данных показал, что алгоритм Storm по сравнению с алгоритмом Streams с увеличением размера изображений потреблял огромное количество памяти в соотношении от 1,3 до 13,5 раз. Точные данные потребления памяти приведены в табл. 1, а на рис. 4 они представлены в графической форме. Это послужило причиной более тщательного анализа алгоритмов и возможностей для оптимизаций по потреблению памяти.

Табл. 1. Потребление памяти для тестовых наборов из 1000 квадратных изображений

Размер стороны изображения (пиксели)	*Используемая память (Мегабайты)*	
	Storm	*IBM Streams*
50	220	164
100	280	166
200	323	164
400	270	166
600	970	169
800	1300	170
1000	2200	177
1200	2700	199

Узким местом в первой реализации топологии Storm оказался оператор записи «writer». Следствием этого стало повышенное потребление памяти из-за интенсивного заполнения изображениями очередей сообщений. Одним из возможных решений проблемы было ограничение размеров очередей до минимального количества, равного двум элементам [29]. Это позволило снизить потребление памяти в «куче» для наиболее крупного изображения из тестируемых до

340 мегабайт. Альтернативным подходом было увеличение параллелизма оператора «writer» до двух, что привело к расходу в «куче» до 300 мегабайт.

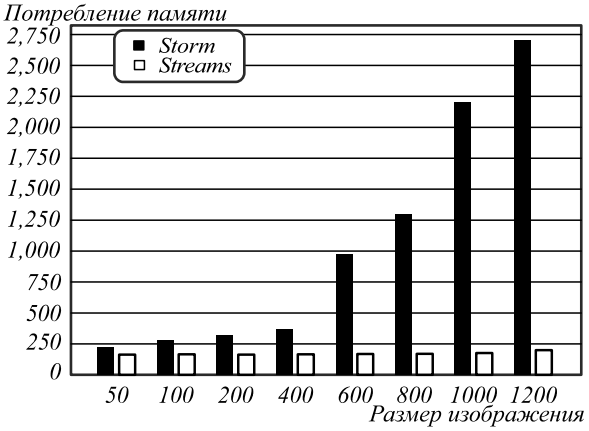

Рис. 4. Использование памяти (МБ) системами Storm и Streams до применения оптимизации

Оптимальной оказалась комбинированная конфигурация с размером очереди, равным 4 элементам, и значением параллелизма оператора «writer», равным 2. В дополнение к этому было установлено ограничение размера «кучи» JVM рабочих процессов в 200 МБ, что позволило без заметных для производительности затрат производить автоматическую сборку мусора.

Алгоритм Streams также оптимизировался, но по пропускной способности, однако улучшить результаты не удалось. После добавления к первым двум операторам конвейера («DirectoryScan» и «ReadImage») оператора применения оконной функции «Smooth» происходило падение пропускной способности. Увеличение степени параллелизма при помощи оператора «ThreadedSplit» приводило к тому же эффекту. Инструменты, предоставляемые вместе со средой разработки IBM Streams, не позволили выявить причину такого поведения.

Основные результаты приведены в табл. 2, 3. На рис. 5 изображён сравнительный график потребления памяти после оптимизации.

Табл. 2. Потребление памяти после оптимизации для тестовых наборов из 1000 квадратных изображений

Размер стороны изображения (пиксели)	*Используемая память (Мегабайты)*	
	Storm	*IBM Streams*
50	220	165
100	250	166
200	284	166
400	310	166
600	320	169
800	400	170
1000	460	178
1200	580	200

Как видно, алгоритм Storm был более требовательным к памяти на всех размерах тестовых изображений, причём рост потребления памяти при увеличении размера обрабатываемых изображений был более явным по сравнению с алгоритмом для системы Streams. Максимальное отношение потребления памяти системами Storm к Streams составило 2,9.

Табл. 3. Пропускная способность после оптимизации для тестовых наборов из 1000 квадратных изображений

Размер стороны изображения (пиксели)	*Пропускная способность (количество обработанных изображений в секунду)*	
	Storm	*IBM Streams*
50	1002	1186
100	730	611
200	202	210
400	108	42,3
600	49,6	23,6
800	32,5	14,5
1000	24	9,51
1200	14,4	5,91

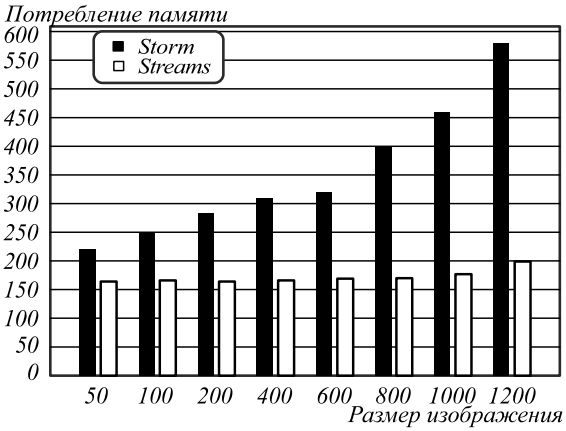

Рис. 5. Использование памяти (МБ) системами Storm и Streams после оптимизации

Побочным результатом оптимизации по памяти стало и увеличение пропускной способности программы Storm (см. табл. 3). Алгоритм оказался более быстрым для изображений с размерами сторон больше 200 и превосходил алгоритм Streams в среднем в 2,3 раза. До оптимизации это значение было равно 1,5. Системы имели примерно одинаковую пропускную способность при обработке небольших изображений, но дополнительные тесты всё-таки показали, что отношение пропускной способности Streams к Storm увеличивается с дальнейшим уменьшением изображений.

На рис. 6, 7 представлены результаты замеров пропускной способности в графической форме. На первом рисунке заметно отсутствие устойчивого превосходства одной из систем при обработке наборов изображений с размерами сторон от 50 до 200 пикселей. Второй рисунок демонстрирует преимущество системы Storm по пропускной способности для больших изображений.

Выводы

Тестирование на виртуальных машинах показало сильные и слабые стороны систем Storm и Streams в поставленной задаче обработки потока изображений.

Storm был быстрее для всех тестовых размеров изображений, кроме небольших – со стороной меньшей 200 пикселей. Превосходство пропускной спо-

собности Storm по отношению к системе Streams составляло от 2 до 2,5 раз.

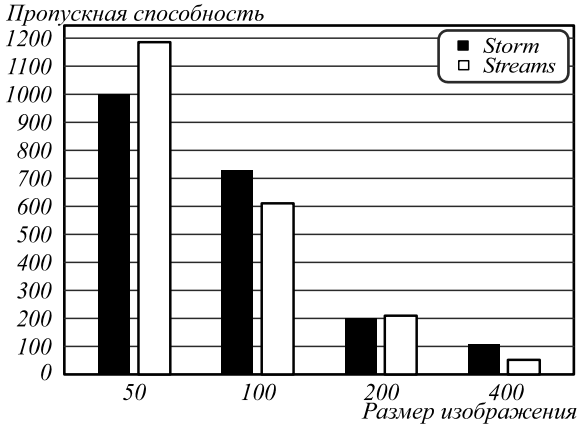

Рис. 6. Пропускная способность систем Storm и Streams на тестовом наборе из 1000 квадратных изображений с размерами сторон от 50 до 400 пикселей

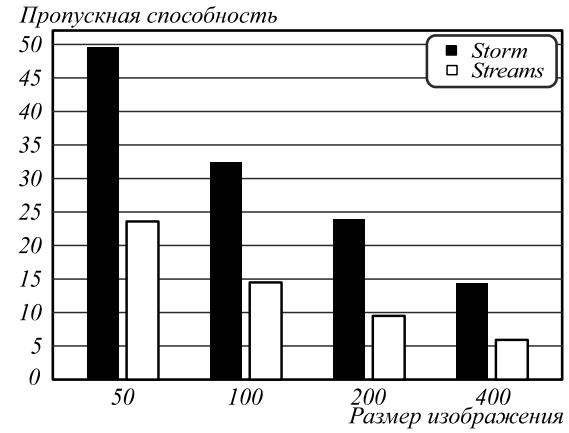

Рис. 7. Пропускная способность систем Storm и Streams на тестовом наборе из 1000 квадратных изображений с размерами сторон от 600 до 1200 пикселей

Система IBM Streams была экономичней по количеству потребляемой памяти (используемые объёмы от 150 до 200 мегабайт по сравнению с диапазоном от 220 до 2700 мегабайт у Storm). Однако в оптимизированной по потреблению памяти версии для Storm удалось снизить верхнюю границу до показателей Streams, что к тому же привело и к увеличению пропускной способности, но потребовало больше времени на разработку.

Таким образом, можно сделать вывод о том, что в локальной конфигурации при отсутствии большого количества оперативной памяти применение IBM InfoSphere Streams будет более выигрышным. В тех случаях, когда требуется получить высокую производительность, следует выбирать Storm, заплатив за это большим в 3-4 раза потреблением оперативной памяти.

Предполагается, что данная работа будет продолжена в направлении усложнения конфигурации тестируемых аппаратных платформ.

Благодарности

Исследование выполнено за счёт Российского научного фонда (РНФ), проект № 14-31-00014.

Литература

1. **Казанский, Н.Л.** Распределённая система технического зрения регистрации железнодорожных составов / Н.Л. Казанский, С.Б. Попов // Компьютерная оптика. – 2012. – Т. 36, № 3. – С. 419-428.
2. **Журавель, Ю.Н.** Особенности обработки гиперспектральных данных дистанционного зондирования при решении задач мониторинга окружающей среды / Ю.Н. Журавель, А.А. Федосеев // Компьютерная оптика. – 2013. – Т. 37, № 4. – С. 471-476.
3. **Kazanskiy, N.L.** Machine Vision System for Singularity Detection in Monitoring the Long Process / N.L. Kazanskiy, S.B. Popov // Optical Memory and Neural Networks (Information Optics). – 2010. – Vol. 19, Issue 1. – P. 23-30.
4. **Гашников, М.В.** Иерархическая сеточная интерполяция при сжатии гиперспектральных изображений / М.В. Гашников, Н.И. Глумов // Компьютерная оптика. – 2014. – Т. 38, № 1. – С. 87-93.
5. **Зимичев, Е.А.** Пространственная классификация гиперспектральных изображений с использованием метода кластеризации k-means++ / Е.А. Зимичев, Н.Л. Казанский, П.Г. Серафимович // Компьютерная оптика. – 2014. – Т. 38, № 2. – С. 281-286.
6. **Попов, С.Б.** Концепция распределённого хранения и параллельной обработки крупноформатных изображений / С.Б. Попов // Компьютерная оптика. – 2007. – Т. 31, № 4. – С. 77-85.
7. **Казанский, Н.Л.** Использование инфраструктуры облачных вычислений для моделирования сложных нанофотонных структур / Н.Л. Казанский, П.Г. Серафимович // Компьютерная оптика. – 2011. – Т. 35, № 3. – С. 320-328.
8. **Kazanskiy, N.L.** Cloud Computing for Rigorous Coupled-Wave Analysis / N.L. Kazanskiy, P.G. Serafimovich // Advances in Optical Technologies. – 2012. – Vol. 2012. – Article ID 398341, 7 pages. – Doi:10.1155/2012/398341.
9. **Kazanskiy, N.L.** Cloud Computing for Nanophotonic Simulations / N.L. Kazanskiy, P.G. Serafimovich // Lecture Notes in Computer Science. – 2013. – Vol. 7715. – P. 54-67. – DOI: 10.1007/978-3-642-38250-5
10. **Волотовский, С.Г.** Оценка производительности приложений параллельной обработки изображений / С.Г. Волотовский, Н.Л. Казанский, С.Б. Попов, П.Г. Серафимович // Компьютерная оптика. – 2010. – Т. 34, № 4. – С. 567-573.
11. **Попов, С.Б.** Моделирование информационной структуры параллельной обработки изображений / С.Б. Попов // Компьютерная оптика. – 2010. – Т. 34, № 2. – С. 231-242.
12. **Pereira, R.** An architecture for distributed high performance video processing in the cloud / M. Azambuja, K. Breitman, M. Endler // Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. – IEEE, 2010. – P. 482-489.
13. **Almeer, M.H.** Cloud Hadoop map reduce for remote sensing image analysis / M.H. Almeer // Journal of Emerging Trends in Computing and Information Sciences. – 2012. – Vol. 3, Issue 4. – P. 637-644.
14. **Perry, R.** High Speed Raster Image Streaming For Digital Presses Using the Hadoop File System [Electronical resource] / R. Perry // HP Laboratories, HPL-2009-345. – 2009. – URL: http://www.hpl.hp.com/techreports/2009/HPL-2009-345.html.
15. **Maxwell, D.** Crisees: Real-time monitoring of social media streams to support crisis management / D. Maxwell, S. Raue, L. Azzopardi, C. Johnson and S. Oates // Advances in Information Retrieval. – Springer Berlin Heidelberg, 2012. – P. 573-575.
16. **Nabi, Z.** Benchmark «Of Streams and Storm» / E. Bouillet, A. Bainbridge, C.Thomas [Электронный ресурс]. – 2014.

– URL: https://developer.ibm.com/streamsdev/wp-content/uploads/sites/15/2014/04/Streams-and-Storm-April-2014-Final.pdf (дата обращения 12.08.2014).
17. Real-time Stream Processing Architecture for Comcast IP Video [Электронный ресурс]. – URL: http://strataconf.com/stratany2013/public/schedule/detail/30915\%29 (дата обращения 12.08.2014).
18. **Глумов, Н.И.** Применение полиномиальных базисов для обработки изображений в скользящем окне / Н.И. Глумов, В.В.Мясников, В.В. Сергеев // Компьютерная оптика. – 1995. – Вып.14-15. – Ч. 1. – С. 55-67.
19. **Glumov, N.I.** Detection of objects on the image using a sliding window mode / N.I. Glumov, E.I. Kolomiyetz, V.V. Sergeyev // Optics & Laser Technology. – 1995. – Vol. 27(4). – P. 241-249.
20. Методы компьютерной обработки изображений / М.В. Гашников, Н.И. Глумов, Н.Ю. Ильясова, В.В. Мясников, С.Б. Попов, В.В. Сергеев, В.А. Сойфер, А.Г. Храмов, А.В. Чернов, В.М. Чернов, М.А. Чичёва, В.А. Фурсов. – под ред. В.А. Сойфера. – 2-е изд., испр. – М.: Физматлит, 2003. – 784 с.
21. Zookeeper [Электронный ресурс]. – URL: http://zookeeper.apache.org/ (дата обращения 12.08.2014).
22. ZeroMQ [Электронный ресурс]. – URL: http://zeromq.org/ (дата обращения 3.08.2014).
23. Фреймворк Netty [Электронный ресурс]. – URL: http://zeromq.org/ (дата обращения 3.08.2014).
24. LMax Disruptor [Электронный ресурс]. – URL: http://disruptor.googlecode.com/files/Disruptor-1.0.pdf (дата обращения 3.08.2014).
25. Протокол Thrift [Электронный ресурс]. – URL: http://thrift.apache.org/ (дата обращения 7.08.2014).
26. Библиотека обработки изображений OpenCV [Электронный ресурс]. – URL: http://opencv.org/ (дата обращения 1.08.2014).
27. Набор операторов OpenCV Toolkit [Электронный ресурс]. – URL: http://github.com/ejpring/OpenCVToolkit (дата обращения 8.08.2014).
28. **Hirzel, M.** IBM Streams Processing Language: Analyzing Big Data in motion / M. Hirzel, H. Andrade, B. Gedik, G. Jacques-Silva, R. Khandekar, V. Kumar, M. Mendell, H. Nasgaard, S. Schneider, R. Soule´ and K.-L. Wu // IBM Journal of Research and Development. – 2013. – Vol. 57. – Issue 3/4. – P. 7:1-7:11. – DOI: 10.1147/JRD.2013.2243535.
29. **Hirzel, M.** A catalog of stream processing optimizations / M. Hirzel, R. Soulé, S. Schneider, B. Gedik, R. Grimm // ACM Computing Surveys (CSUR). – 2014. – Vol. 46, Issue 4. – Article Number: 46. – DOI: 10.1145/2528412.

References

1. **Kazanskiy, N.L.** The distributed vision system of the registration of the railway train / N.L. Kazanskiy, S.B. Popov // Computer Optics. – 2012. – Vol. 36(3). – P. 419-428.
2. **Zhuravel, Yu.N.** The features of hyperspectral remote sensing data processing under environment monitoring task solution / Yu.N. Zhuravel, A.A. Fedoseev // Computer Optics. – 2013. – Vol. 37(4). – P. 471-476.
3. **Kazanskiy, N.L.** Machine Vision System for Singularity Detection in Monitoring the Long Process / N.L. Kazanskiy, S.B. Popov // Optical Memory and Neural Networks (Information Optics). – 2010. – Vol. 19, Issue 1. – P. 23-30.
4. **Gashnikov, M.V.** Hierarchical grid interpolation for hyperspectral image compression / M.V. Gashnikov, N.I. Glumov // Computer Optics. – 2014. – Vol. 38(1). – P. 87-93.
5. **Zimichev, E.A.** Spectral-spatial classification with k-means++ particional clustering / E.A. Zimichev, N.L. Kazanskiy, P.G. Serafimovich // Computer Optics. – 2014. – Vol. 38(2). – P. 281-286.
6. **Popov, S.B.** The concept of distributed storage and parallel processing of large-format images / S.B. Popov // Computer Optics. – 2007. – Vol. 31(4). – P. 77-85 – (In Russian).
7. **Kazanskiy, N.L.** Cloud computing for nanophotonics simulations / N.L. Kazanskiy, P.G. Serafimovich // Computer Optics. – 2011. – Vol. 35(3). – P. 320-328. – (In Russian).
8. **Kazanskiy, N.L.** Cloud Computing for Rigorous Coupled-Wave Analysis / N.L. Kazanskiy, P.G. Serafimovich // Advances in Optical Technologies. – 2012. – Vol. 2012. – Article ID 398341, 7 pages. – Doi:10.1155/2012/398341.
9. **Kazanskiy, N.L.** Cloud Computing for Nanophotonic Simulations / N.L. Kazanskiy, P.G. Serafimovich // Lecture Notes in Computer Science. – 2013. – Vol. 7715. – P. 54-67. – DOI: 10.1007/978-3-642-38250-5_7.
10. **Volotovsky, S.G.** Evaluation of the performance of applications in parallel image processing / S.G. Volotovsky, N.L. Kazanskiy, S.B. Popov, P.G. Serafimovich // Computer Optics. – 2010. – Vol. 34(4). – P. 567-573. – (In Russian).
11. **Popov, S.B.** Modeling information structure of parallel image processing / S.B. Popov // Computer Optics. – 2010. – Vol. 34(2). – P. 231-242. – (In Russian).
12. **Pereira, R.** An architecture for distributed high performance video processing in the cloud / M. Azambuja, K. Breitman, M. Endler // Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. – IEEE, 2010. – P. 482-489.
13. **Almeer, M.H.** Cloud Hadoop map reduce for remote sensing image analysis / M.H. Almeer // Journal of Emerging Trends in Computing and Information Sciences. – 2012. – Vol. 3, Issue 4. – P. 637-644.
14. **Perry, R.** High Speed Raster Image Streaming For Digital Presses Using the Hadoop File System [Electronical resource] / R. Perry // HP Laboratories, HPL-2009-345. – 2009. – URL: http://www.hpl.hp.com/techreports/2009/HPL-2009-345.html.
15. **Maxwell, D.** Crisees: Real-time monitoring of social media streams to support crisis management / D. Maxwell, S. Raue, L. Azzopardi, C. Johnson and S. Oates // Advances in Information Retrieval. – Springer Berlin Heidelberg, 2012. – P. 573-575.
16. **Nabi, Z.** Benchmark «Of Streams and Storm» / E. Bouillet, A. Bainbridge, C.Thomas [Электронный ресурс]. – 2014. – URL: https://developer.ibm.com/streamsdev/wp-content/uploads/sites/15/2014/04/Streams-and-Storm-April-2014-Final.pdf (дата обращения 12.08.2014).
17. Real-time Stream Processing Architecture for Comcast IP Video [Электронный ресурс]. – URL: http://strataconf.com/stratany2013/public/schedule/detail/30915\%29 (дата обращения 12.08.2014).
18. **Glumov, N.I.** Application of polynomial bases for image processing in a sliding window / N.I. Glumov, V.V. Myasnikov, V.V. Sergeyev // Computer Optics. – 1995. – Vol. 14-15, Part 1. – P. 55-67. – (In Russian).
19. **Glumov, N.I.** Detection of objects on the image using a sliding window mode / N.I. Glumov, E.I. Kolomiyetz, V.V. Sergeyev // Optics & Laser Technology. – 1995. – Vol. 27(4). – P. 241-249.
20. Methods for computer image processing / M.V. Gashnikov, N.I. Glumov, N.Yu. Ilyasova, V.V. Myasnikov, S.B. Popov, V.V. Sergeev, V.A. Soifer, A.G. Khramov, A.V. Chernov, V.M. Chernov, M.A. Chicheva, V.A. Fursov; – ed. by V.A. Soifer. – 2nd ed., revised. – Moscow: “Fizmatlit” Publisher, 2003. – 784 p. – (In Russian).
21. Zookeeper [electronic resource]. – URL: http://zookeeper.apache.org/ (request date 12.08.2014).
22. ZeroMQ [electronic resource]. – URL:http://zeromq.org/ (request date 3.08.2014).

23. The framework Netty [electronic resource]. – URL: http://zeromq.org/ (request date 3.08.2014).
24. LMax Disruptor [electronic resource]. – URL: http://disruptor.googlecode.com/files/Disruptor-1.0.pdf (request date 3.08.2014).
25. Thrift Protocol [electronic resource]. – URL: http://thrift.apache.org/ (access date 7.08.2014).
26. The image processing library OpenCV [electronic resource]. – URL: http://opencv.org/ (request date 1.08.2014).
27. A set of operators OpenCV Toolkit [electronic resource]. – URL: http://github.com/ejpring/OpenCVToolkit (request date 8.08.2014).
28. **Hirzel, M.** IBM Streams Processing Language: Analyzing Big Data in motion / M. Hirzel, H. Andrade, B. Gedik, G. Jacques-Silva, R. Khandekar, V. Kumar, M. Mendell, H. Nasgaard, S. Schneider, R. Soule´ and K.-L. Wu // IBM Journal of Research and Development. – 2013. – Vol. 57. – Issue 3/4. – P. 7:1-7:11. – DOI: 10.1147/JRD.2013.2243535.
29. **Hirzel, M.** A catalog of stream processing optimizations / M. Hirzel, R. Soulé, S. Schneider, B. Gedik, R. Grimm // ACM Computing Surveys (CSUR). – 2014. – Vol. 46, Issue 4. – Article Number: 46. – DOI: 10.1145/2528412.

COMPARISON OF SYSTEM PERFORMANCE FOR STREAMING DATA ANALYSIS IN IMAGE PROCESSING TASKS BY SLIDING WINDOW

N.L. Kazanskiy, V.I. Protsenko, P.G. Serafimovich
Samara State Aerospace University,
Image Processing Systems Institute, Russian Academy of Sciences

Abstract

Comparison of two streaming systems, Apache Storm and IBM InfoSphere Streams, was performed when solving a problem of image filtering by a sliding window. The analysis was conducted for two parameters: throughput and memory consumption. Testing was performed under CentOS operating systems running on two virtual machines for each system. First, the most cheap to develop realizations were compared. Following the analysis, optimization of memory consumption was performed. Suggestions on the prospective application areas of two systems were made in the Conclusion.

Key words: big data, stream processing, a set of images, a sliding window, used computer memory, image processing.

Сведения об авторах

Сведения об авторе ***Казанский Николай Львович*** *– см. стр. 775 этого номера.*

Проценко Владимир Игоревич, 1991 года рождения. В 2014 году окончил Самарский государственный аэрокосмический университет имени академика С.П. Королёва (СГАУ) со степенью магистра по направлению «Прикладная математика и информатика». Сотрудник научно-исследовательской лаборатории прорывных технологий дистанционного зондирования Земли СГАУ. Области научных интересов: разработка и исследование программных средств распределённой и параллельной обработки крупноформатных изображений, технологии обработки больших данных.

E-mail: *protsenkovi@gmail.com* .

Vladimir Igorevich Protsenko, Engineer of scientific-research laboratory No 97 of Samara State Aerospace University – Advanced Technologies for Remote Sensing laboratory. His areas of research are parallel and distributed image processing, big data technologies.

Серафимович Павел Григорьевич, кандидат физико-математических наук; старший научный сотрудник Института систем обработки изображений РАН, докторант Самарского государственного аэрокосмического университета имени академика С.П. Королёва. Старший научный сотрудник научно-исследовательской лаборатории прорывных технологий дистанционного зондирования Земли СГАУ. Области научных интересов: моделирование и проектирование нанооптических устройств, методы исследования фотонных кристаллов, разработка и исследование программных средств распределённой и параллельной обработки крупноформатных изображений.

E-mail: *serp@smr.ru* .

Pavel Grigorievich Serafimovich, Candidate in Physics and Mathematics; senior researcher at the Image Processing Systems Institute of RAS. Senior researcher at the Breakthrough Technologies for Earth's Remote Sensing laboratory in S.P. Korolyov Samara State Aerospace University (National Research University). His areas of research are nanooptics, simulation and design of photonic crystals, parallel and distributed image processing.

Поступила в редакцию 7 ноября 2014г.