УДК 004.272.2:519.63

В. В. Чуприн, магистр,
О.А. Дмитриева, д-р техн. наук, проф.,
Донецкий национальный технический университет, г. Красноармейск
dmitrieva.donntu@gmail.com

Организация планирования по принципу «кратчайшая задача первая» для Hadoop MapReduce

Работа посвящена вопросам организации планирования при анализе большого количества данных с использованием каркаса параллельной обработки MapReduce. На основе анализа существующих алгоритмов планирования для реализации в рамках многопроцессорной системы предложен вариант, основанный на принципе «кратчайшая задача - первая». Показано, что данный принцип обеспечивает время реакции системы, близкое к оптимальному. Полученные экспериментально результаты, основанные на реальных рабочих нагрузках, генерируемых с помощью стандартного тестового пакета, фиксируют значительное снижение времени реакции системы по отношению к широко используемому стандартному планировщику Hadoop и показывают, что предложенный алгоритм в значительной степени устойчив к ошибкам оценки размера заданий.

Ключевые слова: Hadoop, планировщик, ресурсы, кластер, распределенная система, приоритет, MapReduce.

Введение

Современный этап развития общества характеризуется накоплением огромных объемов информации, обрабатывать и извлекать которую становится все сложнее и дороже [1-2]. В [3] утверждается, что 90% данных, существующих в сегодняшнем мире, создано за последние два года. Библиотека Конгресса США сообщает [4], что ее архив общественных сообщений Twitter достиг 170 миллиардов твитов и растет, примерно, на 500 миллионов твитов в день. Данные поступают из социальных сетей, от датчиков для сбора информации о климате, из публикаций на сайтах социальных сетей, из записей о транзакциях по продажам, от операций по проверке баланса на банковских картах, с цифровых изображений и видеозаписей, формируются как сигналы GPS с мобильных телефонов. Объем данных настолько значителен, что их обработка с помощью стандартных программных и аппаратных средств становится невозможной. Серьезное препятствие для обработки представляет и недостаточная структурированность данных, а также необходимость обрабатывать информацию с высокой скоростью. По оценкам специалистов [4-5], существует так называемая проблема «Big Data», связанная с большим объемами информации, их разнообразием и требованиями относительно времени обработки. Также под этой проблемой понимают определенный набор средств, методов и технологий, позволяющих решить данные задачи. Одним из таких подходов является распределение вычислений, позволяющее при обработке больших объемов данных использовать группу высокопроизводительных машин, объединенных в кластер, а также использование специальных программных каркасов.

Появление задач, требующих анализа больших объемов данных с использованием каркасов параллельной обработки, таких как MapReduce [6], создали потребность в управлении ресурсами вычислительных кластеров, которые работают в разделяемом, многопользовательском окружении. Пользователями системы может быть группа лиц, которые характеризуются такими параметрами, как приоритеты, зарезервированное количество вычислительных мощностей, гарантированное количество разделяемого дискового пространства и пр. Вместе с этим, множество пользователей могут разделять ресурсы одного кластера, так как это предотвращает простои и приводит к значительной экономии средств [7-8]. Изначально спроектированные для нескольких очень больших пакетных заданий распределенные каркасы сейчас используются множеством компаний для решения технологических, экспериментальных и научных задач по аналитике данных.

Постановка задачи

При исследовании больших объемов данных, проведении предварительного анализа и проверке работоспособности алгоритмов обработки на небольших наборах часто возникает потребность в повторяющихся вопросах к оператору. Кроме того, некоторые планировщики рабочего процесса, такие как Oozie [9], способствуют увеличению неоднородности рабочей нагрузки, генерируя ряд сопутствующих задач. Интерактивность и сопутствующие задачи не должны замедлять время обслуживания, даже если в текущий момент запущены большие промышленные задачи [6-8]. Как правило, задача администратора кластера предполагает ручную настройку ряда пулов с целью выделения ресурсов на различные категории работ, а также тонкое конфигурирование параметров, регулирующих распределение ресурсов. Этот процесс является трудоемким, подвержен ошибкам и не может быть легко адаптирован к изменениям в составе рабочей нагрузки.

Именно поэтому предлагаемая работа направлена на решение проблемы планирования заданий путем оптимального распределения ресурсов между параллельными процессами с использованием Hadoop [10], как наиболее распространенной реализации MapReduce с открытым исходным кодом. В рамках решения поставленной задачи предполагается проектирование нового протокола планирования, который реализует как справедливое, так и эффективное распределение ресурсов кластера, направленное на достижение минимального времени отклика системы. Приведенный подход удовлетворяет как интерактивные потребности для маленьких заданий, так и требования к производительности для больших задач, которые могут таким образом сосуществовать в кластере, не прибегая к ручной настройке и сложному конфигурированию.

Цель работы состоит в повышении эффективности реализации модели распределенных вычислений MapReduce при обработке больших объемов данных в рамках программного каркаса Hadoop.

Анализ существующих алгоритмов планирования

Реализация MapReduce в Hadoop во многом соответствует подходу, описанному в [6]. Hadoop запускает несколько Map- задач и несколько Reduce- задач параллельно на каждом подчиненном узле – по две на каждом для перекрытия простоя CPU и дисковой подсистемы. Каждый подчиненный узел посылает главному узлу сообщение, когда слоты для вычисления задач становятся пустыми, что позволяет планировщику назначить новые задания узлам. Встроенный в Hadoop планировщик запускает задачи в порядке «First Input, First Output» (FIFO) с пятью уровнями приоритетов [11]. Когда слоты для выполнения задач освобождаются, планировщик сканирует имеющиеся задачи в порядке убывания приоритетов и сохраняет оценки времени, затраченного на поиск требуемой задачи.

Планировщик «Longest Approximate Time to End» (LATE) обычно спекулятивно запускает задачи, которые будут иметь наименьшую продолжительность выполнения, так как эти задачи предоставляют наилучшие возможности спекулятивного копирования для опережения выполнения оригинальных задач и минимизации времени выполнения задания [12]. Интуитивно подобная «жадная» политика хороша, если узлы работают с постоянной скоростью, а также при отсутствии затрат на запуск задачи на другом простаивающем узле.

Планировщик FAIR scheduler [13] использует 2-х уровневую иерархию. На верхнем уровне выполняется разбиение задач на пулы, а на нижнем разбиение слотов, выделенных под пул. FAIR использует те же алгоритмы для распределения слотов между задачами пула, что и планировщик Hadoop для разбиения на узлы [14]. Таким образом, возможны различные модификации, отличные от FIFO.

При планировании вычислительной мощности в Hadoop 2.0 Capacity scheduler [6] вместо пулов создаются несколько очередей, каждая – с конфигурируемым количеством слотов для Map- и Reduce- фаз [14]. Каждой очереди также назначается некоторая гарантированная вычислительная мощность, при этом общая вычислительная мощность кластера представляет собой сумму вычислительных мощностей всех очередей. Очереди контролируются, и если какая-либо очередь не потребляет выделенную ей вычислительную мощность, излишки могут быть временно отданы другим очередям.

Реализация системы с рассмотренными проколами планирования порождает целый ряд нерешенных проблем. Некоторые из них исходят из того, что задачи MapReduce планируются путем простого разбиения задания на подзадачи, при том, что планировщики, как правило, не имеют сведений о размерности обрабатываемых задач.

Разработка алгоритма планирования на основе оценивания размера задания

В работе для оценки размерности задания планировщиком предлагается введение

дополнительных модулей, один из которых до момента начала выполнения оценивает размер задания приблизительно, а далее уточняет его после того, как отработают несколько шаблонных заданий. Полученная оценка дополнительно обрабатывается модулем старения, который формирует приоритеты заданий. Наконец, планировщик использует полученные приоритеты для выделения ресурсов, в то время, как происходит проверка с целью минимизации погрешности оценки времени выполнения задания.

В основе планирования с оценкой размера задания лежит идея повышения приоритетов малых заданий, так как они не будут замедлять работу кластера. Политика «кратчайшая задача - первая», которая определяет приоритеты работы на основе наименьшего времени выполнения, позволяет минимизировать время отклика, то есть время, которое проходит между постановкой задания в очередь и ее завершением [15]. Однако подобный подход может привести к состоянию простоя, если постоянно будут поступать мелкие задачи, тогда более крупные задачи не будут планироваться к выполнению. Для того чтобы этого избежать, в работе предлагается введение показателя «старения» задачи, под которым понимается виртуальное уменьшение размера задания, находящегося в очереди. Этот показатель обеспечивает, в конечном итоге, запуск всех запланированных заданий.

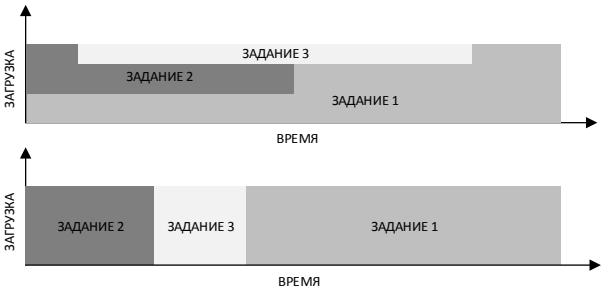

Рисунок 1 – Разделение процессорного времени в планировщике Hadoop (вверху) и планирование по принципу «кратчайшая задача – первая» (внизу).

На рис. 1 приведено сравнение планирования на основе разделения процессорного времени и предложенного подхода. Суммарное время пребывания задач на этапе выполнения, в первом случае выше, чем во втором. Стоит отметить, что время завершения наиболее длительного задания не пострадало от предоставления преимуществ более мелким [16]. В разработанной архитектуре планировщик обрабатывает набор приоритетов, которые являются результатом работы модуля старения. Далее описаны основные задачи, которые были поставлены во время реализации представленного планировщика, а также обоснованы основные архитектурные решения.

Вытеснение низкоприоритетных заданий. В отличие от протоколов, лежащих в основе работы Hadoop, которые планируют задания в целом, приведенный планировщик работает на уровне задач. В общем виде, когда приоритет запущенного задания ниже, чем приоритет одного из находящихся в очереди, запущенное задание должно быть перемещено в очередь ожидания с целью освобождения ресурсов. В Hadoop эта операция может быть реализована либо при помощи прерывания запущенных задач, либо ожидания завершения этих задач. Стоит отметить, что результаты планирования более критичны при высокой загрузке кластера, и прерывание обработки текущих задач может привести к увеличению уровня загруженности, так как работа, уже выполненная прерванными заданиями, должна быть проделана заново. В связи с этим в рамках статьи предложен планировщик, обеспечивающий завершение обработки текущих заданий вместо прерывания.

Фазы задания. В MapReduce задание разбивается на Map- фазу, за которой опционально может следовать фаза Reduce. Предложенный алгоритм оценивает размер задания на основе времени, требуемого на выполнения нескольких первых задач для каждой фазы. В связи с этим невозможно оценить время выполнения фазы Reduce во время выполнения Map- задач. В целях выполнения планирования Map- и Reduce- фазы объединяются в два отдельных задания для планировщика. При планировании на основе минимального времени выполнения, оптимальное время достигается за счет получения оптимальных оценок отдельно для наборов Map- и Reduce- задач.

Модуль обучения. Изначально модуль оценивания предоставляет «грубую» оценку времени выполнения нового задания. Далее оценка корректируется на основе времени выполнения нескольких тестовых заданий. Для гарантирования быстрого преобразования временных оценок в актуальные значения планировщик присваивает более высокий приоритет тестовым заданиям, однако в рамках некоторого порога. Данный подход позволяет предотвратить простаивание регулярных заданий на фоне вновь поступивших.

Локальность данных. Для повышения производительности важно убедится, что Map-задачи выполняют обработку данных, находящихся максимально близко. С этой целью

используется стратегия отложенного планирования [17]. Приведенная стратегия откладывает планирование задач с обработкой удаленных данных на протяжении фиксированного количества попыток, отдавая преимущество менее приоритетным заданиям с данными, расположенными ближе. В связи с этим задачи, принадлежащие к заданиям более низкого приоритета, могут быт запланированы ранее.

Политика планирования. Результатом приведенных выше положений стала политика планирования, которая работает следующим образом:

1) Выбираются задания, которые содержат задачи, ожидающие выполнения в соответствии с ограничениями задержки времени планирования.

2) Выбранные задания сортируются в соответствии с приоритетами, полученными от модуля старения.

3) Проверяется, не занимают ли тестовые задания большую часть слотов, чем заданное пороговое значение, а также проверяется необходимость запуска тестовых задач для оценки времени выполнения. Если тестирование необходимо, планируется тестовая задача для наиболее приоритетного задания. Если в тестировании нет необходимости, будет запущена задача с наивысшим приоритетом из планируемых к запуску.

Экспериментальное исследование разработанного планировщика

В подразделе приведен сравнительный анализ производительности предложенного и стандартного Hadoop- планировщика, основанного на дисциплине FIFO.

Тестовое окружение. Для тестирования был использован кластер с двумя узлами TaskTracker с 1 CPU и 512MB памяти. Hadoop был сконфигурирован в соответствии с рекомендациями [18] и имел следующие параметры: размер блока HDFS 128MB с уровнем репликации 2; каждый TaskTracker имел 2 слота для выполнения Map заданий и 1 для Reduce (рис. 2).

Тестовый кластер был развернут в рамках операционной системы Debian 7 Wheezy, работающей под управлением гипервизора VMware Workstation. Использованная реализация виртуальной машины Java - OpenJDK 1.7.0_65.

Рабочие нагрузки. Создание реалистичной рабочей нагрузки для Hadoop требует отдельного внимания. В рамках проделанной работы была использована стандартная утилита оценки производительности SWIT [18]. Данная утилита включает в себя как компоненты для создания тестовых данных, так и для набора выполняемых заданий, работающие по алгоритмам, описанным в [19-20]. Результирующие тестовые данные описываются следующими параметрами:

- интервал поступления новых заданий;
- количество Map- и Reduce- задач, запускаемых в рамках одного задания;
- характеристики задач, которые включают соотношения входящих и результирующих данных в рамках Map- заданий.

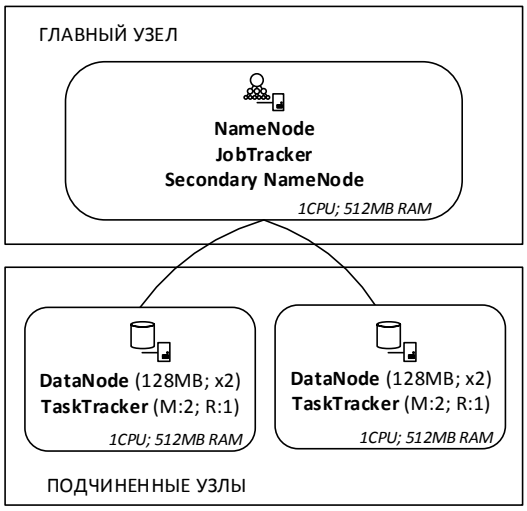

Рисунок 2 – Тестовый кластер

Для проведения экспериментов был использован тестовый набор данных, сгенерированный на основе статистики работы кластеров Facebook. Приведенная характеристика публично доступна под названием FB2009 и подробно описана в [21].

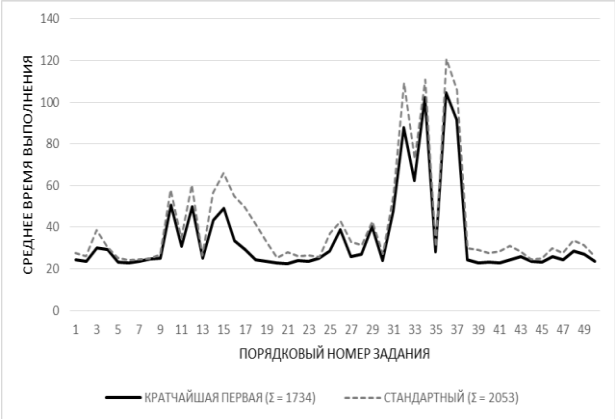

Рисунок 3 – Результаты моделирования

Оценка полученных результатов. Для того, чтобы оценить общую производительность разработанного планировщика, проводилось сравнение временных оценок, полученных в

стандартном планировщике по времени пребывания задания на этапе выполнения (промежутку между постановкой задания в очередь выполнения и его завершением). Оценки строились по 50 заданиям, сгенерированным утилитой оценки производительности, которые запускались по 8 раз для обоих планировщиков. Для сравнения использовались оценки математического ожидания времени выполнения каждого задания разными планировщиками. Полученное распределение времени выполнения приведено на рис. 3. Суммарное среднее время работы заданий в рамках разработанного планировщика составило 1734 секунд против 2053 секунд для выполнения тех же заданий с использованием стандартной дисциплины планирования. Таким образом, был получен 8,4% прирост производительности. На диаграмме распределения времени видно, что максимальный прирост был получен для наиболее длительных заданий. Это дает основания полагать, что при больших нагрузках прирост также увеличится. С целью проверки данной гипотезы, была проведена обработка использованных ранее данных при помощи более ресурсоемкого алгоритма для приведенных выше дисциплин планирования. Результатом данной проверки стал прирост производительности, составивший 8,6%. Такое поведение планировщика объясняется потребностью накопления статистики для предоставления модулем оценки более точных значений.

Выводы

Приведенная работа была мотивирована пониманием того, что парадигма MapReduce эволюционировала до состояния, при котором разделяемые кластеры используются для широкого спектра рабочих нагрузок и включают в себя значительную часть интерактивной обработки данных. При этом преобладающей становится практика развертывания, при которой длительность времени пребывания из-за справедливого распределения ресурсов между конкурирующими заданиями компенсируется более масштабными кластерами. Кроме того, эффективное использование кластера зачастую достигается при помощи утомительного ручного конфигурирования, которое включает создание статического пула ресурсов для обработки всего многообразия рабочих нагрузок, и значительных усилий по настройке.

Для преодоления таких ограничений в работе была предложена новая дисциплина планирования, которая, с одной стороны, отдает предпочтение заданиям с малым временем выполнения, с другой стороны, обеспечивает справедливость распределения ресурсов среди работающих заданий с учетом их приоритетов. Таким образом, был предложен подход, основанный на оценке размеров планируемых заданий в Hadoop. Попутно в работе была реализована процедура оценки размера задания во время выполнения, рассмотрены вопросы предотвращения простаивания как малых, так и больших заданий, что гарантировало сокращение времени пребывания задания в системе. Приведенные проблемы были решены в контексте мультисерверных систем с использованием виртуального времени и старения заданий. Предложенный планировщик был протестирован при помощи стандартной утилиты оценки производительности, которая выполняет генерацию заданий, близких к реальным. Полученные результаты показывают, что разработанное решение чувствительно к широкому спектру нагрузок. Показано, что дисциплина планирования, основанная на оценке размера заданий, успешно справляется с поставленными задачами и обеспечивает существенный прирост производительности. Полученные экспериментальные результаты сравнения приведенной дисциплины планирования и стандартного (FIFO) планировщика демонстрируют как достижение хорошей интерактивности, так и производительности. Как большие, так и малые задания не подвержены простаиванию, а распределение времени выполнения эффективнее в предложенной системе планирования. Разработанный планировщик является практически полезным, так как упрощает процесс конфигурирования и позволяет консолидировать пулы ресурсов. Большое разнообразие рабочих нагрузок эффективно покрывается дисциплиной планирования на основе оценки размера задания.

Дальнейшие исследования будут направлены на разработку подходов планирования, связанных с вытеснением заданий. Перспективным выглядит подход, позволяющий заполнять пробелы между ожиданием и фактическим снятием запущенных задач. Таким образом, дальнейшая цель исследования - реализовать новое множество примитивов для более эффективного приостановления и возобновления выполнения задач с целью обеспечения оптимального вытеснения.

Список использованной литературы

1. Mashey J. R. Big Data ... and the Next Wave of InfraStress [Electronic resource] / J. R. Mashey // Technology Waves. – Access mode: http://static.usenix.org/event/ usenix99/invited_talks/mashey.pdf
2. Sutherland I.E. Logical Effort: Designing Fast CMOS Circuits / I.E. Sutherland, R. F .Sproull, D. F .Harris. – San Francisco: Morgan Kaufmann. – 2009. – 239 p.
3. Maximize Your Relationship [Electronic resource] / IBM Big Data Seminar - Access mode: http://www-304.ibm.com/events/idr/idrevents/detail.action?meid=14730&ieid=7732
4. Miller L. The Origins of «Big Data»: An Etymological Detective Story Credit [Electronic resource] / L. Miller // The New York Times. – Access mode: http://bits. blogs.nytimes.com/2013/02/01/the-origins-of-big-data-an-etymological-detective-story/?_r=0
5. Decisions for the analysis of large volumes of data of IBM PowerLinux Big Data Analytics [Electronic resource] // Technical Paper Search. – Access mode: http://www-03.ibm.com/systems/ru/power/software/linux/powerlinux/bigdata.html
6. Dean J. MapReduce: Simplified data processing on large clusters / J. Dean, S. Ghemawat // Communications of the ACM – 2008. – Vol. 51, №. 1. – P. 107–113.
7. Chen Y. Interactive query processing in big data systems: A cross-industry study of MapReduce workloads [Electronic resource] / Y. Chen, S. Alspaugh, R. Katz] // Technical Paper Search. – Access mode: http://www.eecs.berkeley.edu/~alspaugh/papers/ mapred_workloads_vldb_2012.pdf
8. Hadoop's adolescence: An analysis of Hadoop usage in scientific workloads [Electronic resource] / [Ren K., Kwon Y., Balazinska M., Howe B.] // in Proc. of VLDB. – Access mode: http://homes.cs.washington.edu/~magda/papers/ren-vldb13.pdf
9. Apache Oozie Workflow Scheduler for Hadoop [Electronic resource]. – Access mode: http://oozie.apache.org.
10. Hadoop: Open source implementation of MapReduce [Electronic resource]. – Access mode: http://hadoop.apache.org/.
11. Improving MapReduce Performance in Heterogeneous Environments / [Zaharia M., Konwinski A., Joseph A. D., Katz R., Stoica I.] // Electronic edition. – Berkley, 2009. – P. 69-75.
12. Xiao Z. A Hierarchical Approach to Maximizing MapReduce Efficiency / Z. Xiao, H. Chen, B. Zang // 2011 International Conference on Parallel Architectures and Compilation Techniques. – 2011. – P. 167-168.
13. Zaharia M. Improving MapReduce Performance in Heterogeneous Environments / M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, I. Stoica // Electronic edition. – Berkley. – 2009. – P. 325-331.
14. The Hadoop Distributed File System. / [K. Shvachko, H. Kuang, S. Radia, R. Chansler] //MSST '10 Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). – 2010. – P. 1-10.
15. Таненбаум Э. Операционные системы. Разработка и реализация. / Таненбаум Э., Вудхалл А. // 3-е изд. – СПб.: 2007. — 704 с.
16. Size-based scheduling to improve web performance/ [M. Harchol-Balter, B. Schroeder, N. Bansal, M. Agrawal] // School of Computer Science Carnegie Mellon University Pittsburgh. – 2003. – Vol. 21, №. 2. – P. 1–19.
17. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling/ [M. Zaharia, D. Borthakur, J. S. Sarma et al.] // Proceedings of the 5th European conference on Computer systems. – 2010. – P. 265-278.
18. Statistical workload injector for mapreduce/ [Electronic resource] // Project at UC Berkeley AMP Lab. – Access mode: https://github.com/SWIMProjectUCB/SWIM.
19. Y. Chen. We don't know enough to make a big data benchmark suite - an academia-industry view [Electronic resource] / Chen Y. // In Proc. Of WBDB, 2012. – Access mode: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.357.4593.
20. Чуприн В.И. Исследование алгоритмов планирования MapReduce заданий на кластерах Hadoop / В.И. Чуприн, О.А. Дмитриева // Збірка матеріалів V Всеукраїнської науково - технічної конференції студентів, аспірантів та молодих вчених «Інформаційні управляючі системи та комп'ютерний моніторінг», Донецьк , 24-25 квітня 2014. — Донецьк: ДонНТУ, 2014. — Т. 2. – С. 168-174.
21. The case for evaluating mapreduce performance using workload suites / [Y. Chen, A. Ganapathi, R.Griffith, R. Katz.]// IEEE MASCOTS. – Berkeley. – 2011. – P. 390–399.

Надійшла до редакції 01.03.2015

В.І. ЧУПРІН, О.А. ДМИТРІЄВА
Донецький національний технічний університет, м. Красноармійськ

ОРГАНІЗАЦІЯ ПЛАНУВАННЯ ЗА ПРИНЦИПОМ «НАЙКОРОТША ЗАДАЧА ПЕРША» ДЛЯ HADOOP MAPREDUCE

Робота присвячена питанням організації планування при аналізі великої кількості даних з використанням каркаса паралельної обробки MapReduce. На основі аналізу існуючих алгоритмів планування для реалізації в рамках багатопроцесорної системи запропонований варіант, заснований на принципі «найкоротша задача перша». Показано, що даний принцип забезпечує час реакції системи, близький до оптимального. Отримані експериментально результати, засновані на реальних робочих навантаженнях, що генеруються за допомогою стандартного тестового пакета, фіксують значне зниження часу реакції системи стосовно широко використовуваного стандартного планувальника Hadoop і показують, що запропонований алгоритм у значній мірі стійкий до помилок оцінки розміру завдань.

Ключові слова: Hadoop, планувальник, ресурси, кластер, розподілена система, пріоритет, MapReduce.

V. I. CHUPRYN, O.A. DMITRIEVA
Donetsk National Technical University, Krasnoarmiysk

ORGANIZATION OF PLANNING BY THE PRINCIPLE "THE SHORTEST FIRST TASK" FOR HADOOP MAPREDUCE

The work considers the organization of planning in the analysis of large amounts of data with the use of parallel processing carcass MapReduce. On the basis of the analysis of existing scheduling algorithms for implementation within the processor system we offer an option based on the principle "The shortest first task". It is shown that this principle provides a response time that is close to optimal. This article proposes a scheduler that uses the described technique of planning based on the size of the tasks for intensive data processing systems in a distributed and widely used system of Hadoop. Planning based on the size of the task requires a priori information about the size of the problem, which is absent in Hadoop. The above scheduler receives the required estimate by collecting statistics for the entire period of the cluster work. The given problems were solved in the context of multiserver systems with the use of virtual time and aging of tasks. The developed scheduler is useful as it simplifies the process of configuration and allows consolidating pools of resources. A big variety of working loadings effectively becomes covered by discipline of planning on the basis of task size assessment.

The experimental results based on real workloads generated by a standard test suite record a significant reduction in the response time of the system in relation to the widely used standard Hadoop scheduler and show that the proposed algorithm is largely resistant to job size estimation errors.

Key words: Hadoop, scheduler, resources, cluster, distributed system, priority, MapReduce.