РЕАЛИЗАЦИЯ ПЛАТФОРМЫ РАСПРЕДЕЛЕННЫХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ СБОРА И ПРЕДВАРИТЕЛЬНОЙ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ МОНИТОРИНГА В КИБЕР-ФИЗИЧЕСКИХ СИСТЕМАХ

И. Б. Саенко, А. Г. Кушнеревич, И. В. Котенко

Санкт-Петербургский институт информатики и автоматизации Российской академии наук (СПИИРАН), Санкт-Петербург, Россия
e-mail: ibsaen@mail.ru

Рассматриваются вопросы построения программной системы сбора и предварительной обработки больших массивов гетерогенных данных в кибер-физических системах. Система основана на платформе распределенных параллельных вычислений Hadoop и функционирующей на ее основе системе Spark. Реализация платформы позволяет реализовать потоковую обработку собираемых данных мониторинга на основе технологии Complex Event Processing и способствует преодолению ограничений при обработке Больших Данных. Экспериментальная оценка показала высокую производительность платформы.

Ключевые слова: параллельные вычисления; Большие Данные; мониторинг; кибер-физическая система.

IMPLEMENTATION OF THE DISTRIBUTED PARALLEL COMPUTING FRAMEWORK FOR COLLECTING AND PREPROCESSING THE BIG MONITORING DATA IN CYBER-PHYSICAL SYSTEMS

I.B. Saenko, A.G. Kushnerevich, I.V. Kotenko

St. Petersburg institute of informatics and automation of the Russian Academy of Sciences (SPIIRAS)
St. Petersburg, Russia

The paper considers features of creating the program system of collection and preprocessing big arrays of heterogeneous data in cyber-physical systems. The system is based on a platform of the distributed parallel computing of Hadoop and the Spark system functioning on its basis. Implementation of the framework allows realizing stream processing the collected monitoring data on the basis of Complex Event Processing technology and promotes overcoming restrictions when processing Big Data. The experimental assessment showed the high performance of the framework.

Keywords: parallel computing; Big Data; monitoring; cyber-physical system.

ВВЕДЕНИЕ

Кибер-физические системы (КФС) рассматриваются многими исследователями как одно из наиболее перспективных направлений дальнейшего развития информационно-коммуникационных систем. Важными особенностями КФС являются: 1) очень большое количество источников данных; 2) узлы сети имеют ограниченные вычислительные ресурсы (это особенно справедливо для интернета вещей); 3) для передачи данных от источников к центральным устройствам используются различные типы сетей связи (интернет, мобильные, беспроводные, проводные и т. д.) [1]. Эти особенности делают достаточно актуальной проблему параллельной обработки в КФС вещей больших массивов собираемых данных. Данная проблема объединяет в себе две другие проблемы: обработку Больших Данных и создание масштабируемых вычислительных сетей.

Особенно остро в КФС стоит вопрос мониторинга, который заключается в сборе и предварительной обработке больших массивов гетерогенных данных о событиях и инцидентах [2]. При этом сбор и предварительная обработка данных мониторинга должна выполняться «на лету», путем обработки потоков данных. Для этой цели разработана программная платформа, обеспечивающая распределенную потоковую обработку данных о событиях безопасности на основе технологии Complex Event Processing (CEP) и предназначенная для реализации в КФС. Основные решения, связанные с разработкой этой платформы и отличающие ее от других систем аналогичного типа, касаются следующих аспектов: 1) технологии обработки потоков данных; 2) выбора программно-инструментального средства для реализации платформы; 3) архитектуры платформы.

ТЕХНОЛОГИЯ ОБРАБОТКИ ПОТОКОВ ДАННЫХ

Технология CEP, лежащая в основе обработки потоков данных, предполагает, что над потоками данных можно выполнять некоторую совокупность операций по аналогии, как в реляционных базах данных выполняются операции над таблицами данных [3, 4]. При этом каждый поток данных воспринимается как следующие друг за другом записи (кортежи) данных, имеющие одинаковую *схему*. Схема записей данных является также *схемой потока* данных.

Платформа ориентирована на выполнение следующих потоковых операций: отображения *Map*, фильтрации *Filter*, объединения *Union*, агрегации *Aggregate* и соединения *Join*. Операция *Map* является аналогом реляционной проекции, *Filter* – селекции, *Union* – объединения, *Join* – соединения. Действия, выполняемые операцией *Aggregate*, аналогичны действиям по инструкции GROUP BY в языке SQL.

Пусть для потока S_0 необходимо решить следующую задачу мониторинга: «Выдавать предупреждение каждый раз, когда в течение последних N секунд в потоке будет обнаружено M обращений к заданному порту Z». Тогда для ее решения следует выполнить следующую цепочку операций потоковой обработки данных:

$$S_1 = Filter\left(S_0 \mid Port = Z\right),$$
$$S_2 = Aggregate\left(S_1 \mid w = N,\ A_1 = \mathrm{COUNT}(Port)\right),$$
$$S_3 = Filter\left(S_2 \mid A_1 \geq M\right),$$

где S_1 и S_2 – промежуточные, а S_3 – результирующий потоки; *Port* – атрибут схемы потока S_0, отражающий номер порта; w – размер окна контроля записей потока; A_1 – ат-

рибут промежуточного потока S_2, отражающий результат функции агрегации COUNT применительно к атрибуту *Port*.

ВЫБОР ПРОГРАММНО-ИНСТРУМЕНТАЛЬНОГО СРЕДСТВА ДЛЯ РЕАЛИЗАЦИИ ПЛАТФОРМЫ

Выбор программно-инструментального средства для реализации платформы ориентирован на поддержание выше приведенного набора потоковых операций.

На данный момент стандартом в области распределенных вычислений является система *Hadoop*, хотя в последнее время широкую популярность также приобретает система *Spark* [5]. В силу того, что основой *Hadoop* является вычислительная модель *MapReduce*, любой алгоритм для *Hadoop* состоит из двух методов, а именно: *mapper* и *reducer*. К тому же, необходим еще и третий метод *task*, который запускает задачу. Из-за этого программный код, написанный при подходе *MapReduce*, является громоздким и сложно читаемым.

Система *Spark* использует свою вычислительную модель, основанную на *RDD*-блоках. В результате программный код для *Spark* на порядок короче, чем для *Hadoop*, и состоит всего из одного метода. Кроме того, запись данных на жесткий диск в *Spark* выполняется не так часто, как в *Hadoop* (в *Hadoop* она выполнятся при каждом выполнении операции *Map*). При этом *Spark* больше и эффективнее использует оперативную память машины, что выливается в более высокую скорость работы и выигрыш на итеративных алгоритмах до 100 раз по оперативной памяти и до 10 раз по требуемому пространству на жестком диске. Это определило выбор *Spark* как основного средства, ориентированного на внедрение в КФС.

АРХИТЕКТУРА ПЛАТФОРМЫ

Структура разработанной платформы включает в себя следующие модули: 1) сбора данных; 2) агрегации данных; 3) нормализации и анализа данных; 4) визуализации данных; 5) работы с хранилищем данных. Все модули платформы базируются на средствах реализации параллельных потоковых вычислений Hadoop / Spark, установленных на платформе виртуализации. Этим обеспечивается необходимые мобильность и гибкость платформы, позволяющие настраивать ее под конкретную конфигурацию КФС.

Модуль сбора данных реализует многоагентный подход, при котором на каждом источнике данных (датчике) размещается приложение-агент, взаимодействующее с множеством рабочих серверов и главным сервером модуля сбора данных. Модуль сбора осуществляет предварительную обработку входного файла.

Собранные данные записываются в распределенную файловую систему (HDFS – Hadoop Distributed File System), формируемую средствами Hadoop / Spark. Файловая система HDFS является основой построения модуля хранения данных. Она является альтернативным и более эффективным средством хранения данных, чем традиционная система управления базами данных, а также перспективные решения, связанные с построением онтологического хранилища [6].

После записи собранных данных в HDFS они упаковываются в поток данных с конкретной схемой. Минимальный вариант схемы данных содержит следующие поля: IP-адрес источника; номер порта источника; IP-адрес получателя; номер порта получателя. Данный поток передается далее на модуль агрегации данных.

Модуль агрегации данных обрабатывает поток средствами Hadoop / Spark, используя функционал технологии потоковой обработки CPE, и осуществляет вычисление мер центральной тенденции и экстремумов.

Для модуля агрегации реализованы две схемы распараллеливания процесса агрегации. Если выполняются агрегирующие функции COUNT, SUM, MAX и MIN, то операция *Aggregate* применяется для каждого из подпотоков S_1, S_2, … S_n, на которые делится исходный поток S_0. Полученные результаты затем объединяются с помощью *Union*. Наконец, к объединенному потоку еще раз применяется *Aggregate*. Если выполнятся агрегирующая функция AVERAGE, то схема распараллеливания является более сложной. Это вызвано тем, что среднее значение для выборки **X** при распараллеливании процесса агрегации вычисляется по следующей формуле:

$$\overline{M}(\mathbf{X}) = \sum n_m \overline{M_m} / \sum n_m ,$$

где $\overline{M_m}$ – среднее значение, рассчитанное в m-ой ветви параллельного процесса; n_m – количество обработанных элементов массива данных в m-ой ветви. Поэтому вначале в параллельных ветвях находятся два агрегирующих значения: сумма и количество. Затем, как и в первом случае, применяются операции *Union* и *Aggregate*.

С выхода модуля агрегации поток передается на вход *модуля нормализации и анализа данных*. В данном модуле осуществляется решение более сложных задач по анализу данных. Вначале производится приведение форматов поступающих данных к единому виду, в качестве которого выбран формат CSV (Comma-Separated Values – значения, разделенные запятой). Этот процесс называется *нормализацией*. Затем осуществляется *анализ* данных с помощью предварительно подготовленных правил. Примером является выявление с помощью правил случаев, когда количество пакетов на порт превышает заданное значение. По результату решения этой задачи вырабатываются предупреждающие сообщения и запускаются процедуры контрмер.

С выхода модуля нормализации и анализа данных поток поступает в *модуль визуализации*. В этом модуле осуществляется графическое (визуальное) представление данных мониторинга в соответствии с выбранной моделью. Выбор модели представления осуществляется пользователем. Используются различные стандартизованные и нестандартизованные модели визуализации. К стандартизованным моделям визуализации относятся: гистограммы; круговые диаграммы; линейные графики. К нестандартизованным моделям визуализации относятся: графы; графы, дополненные глифами; карты деревьев; матрицы.

Исходные данные для визуального представления помещаются в базу данных модуля визуализации.

Модуль работы с хранилищем данных обеспечивает работу с HDFS. HDFS обеспечивает высокую оперативность и отказоустойчивость обработки больших массивов данных. Для этой цели реализован механизм репликации блоков данных.

HDFS виртуально объединяет дисковое пространство разных машин в единое адресное пространство для удобства централизованного использования памяти.

Для обеспечения централизованного управления HDFS на главном узле платформы размещается менеджер, главной задачей которого является управление дисковым пространством РФС.

ЭКСПЕРИМЕНТАЛЬНАЯ ОЦЕНКА

Экспериментальная оценка разработанной платформы производилась на тестовых данных, которые в pcap-формате отражают трафик компьютерной сети и загру-

жаются с адреса http://www.fukuda-lab.org/mawilab/v1.1/2015/07/01/20150701.html. Оценивались конфигурации, имеющие 3, 5 и 7 центральных серверов, которые обрабатывали pcap-файлы объемом в 512 Мб, 1 Гб и 1,5 Гб. Полученные времена обработки файлов лежат в диапазоне от 100 секунд (512 Мб на 7 серверах) до 1100 секунд (1,5 Гб на 3 серверах). Это говорит о достаточно высокой производительности разработанной платформы.

ЗАКЛЮЧЕНИЕ

Разработанная программная платформа имеет модульную архитектуру, базируется на Hadoop / Spark и позволяет реализовать эффективную параллельную потоковую обработку гетерогенных данных мониторинга КФС за счет использования встроенных в нее функциональных возможностей технологии CEP. Модули платформы осуществляют сбор, хранение, агрегацию, нормализацию, анализ и визуализацию больших данных мониторинга КФС. Экспериментальная оценка подтвердила высокую производительность, приемлемую ресурсоемкость и пригодность разработанной платформы к использованию в КФС.

Работа выполнена при финансовой поддержке РФФИ (проекты №14-07-00697, 14-07-00417, 15-07-07451, 16-37-00338, 16-29-09482 офи_м), при частичной поддержке бюджетных тем № 0073-2015-0004 и 0073-2015-0007, а также гранта РНФ 15-11-30029 в СПИИРАН.

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1. Нейросетевой подход к прогнозированию состояния элементов сети Интернет вещей / И. В. Котенко [и др.] // Сборник докладов XVIII Междунар. конф. по мягким вычислениям и измерениям (SCM'2015). СПб. : СПбГЭТУ «ЛЭТИ», 2015. Том 1. С. 395–399.

2. Применение технологии управления информацией и событиями безопасности для защиты информации в критически важных инфраструктурах / И. В. Котенко [и др.] // Труды СПИИРАН. СПб. : Наука, 2012. Вып. 1(20). С. 27–56.

3. Anicic D. Stream Reasoning and Complex Event Processing in ETALIS / D. Anicic, S. Rudolph, P. Fodor, N. Stojanovic // Semantic Web, vol. 3, no. 4, 2012. P. 397–407.

4. Gulisano V. StreamCloud: A Large Scale Data Streaming System / V. Gulisano, R. Jimenez-Peris, M. Patino-Martinez, P. Valduriez // Proceedings of the 2010 International Conference on Distributed Computing Systems, 2010. P. 126–137.

5. Spark: Cluster Computing with Working Sets / M. Zaharia [et al.] // Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (HotCloud'10), 2010. P. 10–17.

6. Kotenko I., Polubelova O., Saenko I. The Ontological Approach for SIEM Data Repository Implementation // 2012 IEEE International Conference on Green Computing and Communications, Conference on Internet of Things, and Conference on Cyber, Physical and Social Computing. Besancon, France, November 20-23, 2012. Los Alamitos, California. IEEE Computer Society. 2012. P. 761–766.