УДК 004.056 **DOI** 10.15622/sp.59.1

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

АРХИТЕКТУРА СИСТЕМЫ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ ДЛЯ МОНИТОРИНГА БЕЗОПАСНОСТИ СЕТЕЙ ИНТЕРНЕТА ВЕЩЕЙ

Котенко И.В., Саенко И.Б, Кушнеревич А.Г. **Архитектура системы параллельной обработки больших данных для мониторинга безопасности сетей Интернета вещей.**

Аннотация. Сети Интернета вещей в настоящее время находят свое применение во многих областях жизни людей. Краеугольным камнем в вопросе возможности дальнейшего распространения и использования таких сетей является аспект обеспечения их безопасности. Однако особенности сетей данного вида таковы, что использование в них традиционных средств и систем компьютерной защиты затруднено или невозможно. Одной из таких особенностей является необходимость в режиме реального времени и с минимальными вычислительными затратами анализировать очень большие объемы данных, разнородных по своей природе. С учетом особенностей вычислительных мощностей сети Интернета вещей предлагается архитектура системы параллельной обработки больших данных, основанная на использовании технологии обработки потоков данных Complex Event Processing и платформы параллельных вычислений Hadoop. Рассматриваются вопросы, непосредственно связанные с архитектурой системы, а также с реализацией следующих ее основных компонентов: сбора данных, хранения данных, нормализации и анализа данных и визуализации данных. Взаимосвязь между компонентами обеспечивается с помощью распределенной файловой системы Hadoop, которая является основой для построения распределенного хранилища данных. Компонент сбора данных организует распределенный прием данных и их хранение в компоненте хранилища данных. Компонент нормализации и анализа данных преобразует их к единому формату и обрабатывает с помощью правил корреляции. Компонент визуализации данных представляет данные в графическом виде, более удобном для дальнейшего восприятия оператором. Обсуждаются результаты экспериментальной оценки производительности системы, подтверждающие вывод о ее высокой эффективности.

Ключевые слова: Интернет вещей, мониторинг безопасности, Большие Данные, Complex Event Processing, Hadoop.

1. Введение. Сети Интернета вещей (Internet of Things — IoT) позволяют интегрировать в единую информационную инфраструктуру с помощью различных видов коммуникаций (Интернет, мобильные сети, локальные сети и т.д.) разнообразные компьютерные устройства: центральные компьютеры, пользовательские устройства со встроенными контроллерами, датчики информации об окружающей среде, сенсоры и так далее. По этой причине сети IoT в настоящее время находят все более широкое распространения во многих областях: здравоохранение, транспортные системы, умные дома, робототехника и так далее. При этом сети IoT, в отличие от традиционных компьютерных сетей, имеют следующие особенности: очень большое количество источников данных; очень большой входной поток разнородных данных; узлы сети IoT, как правило, имеют ограниченные сетевые, вычислительные и энергети-

ческие ресурсы. При этом основным проблемным вопросом являются вычислительные ограничения. Необходимо обеспечить либо быстрые вычисления непосредственно на узле сети IoT, либо пересылку данных по сети для анализа на более мощных в вычислительном плане узлах. Эти особенности делают проблему безопасности сетей IoT достаточно острой во всех вышеперечисленных областях их применения.

Традиционные методы и средства защиты информации в сетях IoT являются неэффективными, что вызвано малой мощностью вычислительных ресурсов элементов сетей IoT и большим количеством различных типов используемых в них сетей связи. По этой причине особую актуальность для обеспечения безопасности сетей IoT приобретает подход, связанный с созданием и применением в них интеллектуальных систем управления информацией и событиями безопасности (Security Information and Event Management — SIEM) [1-4]. SIEM-системы осуществляют мониторинг безопасности сети, заключающийся в сборе и предварительной обработке данных о событиях безопасности от удаленных устройств, датчиков информации и элементов сетевой инфраструктуры [1]. Многообразие типов источников данных, используемых для мониторинга сетевой безопасности, и высокая интенсивность формируемых ими потоков событий приводит к необходимости разработки решений, относящихся к проблеме обработки Больших данных. Одним из таких решений является разработка и реализация в сети IoT предлагаемой в настоящей статье системы параллельной обработки данных, предназначенной для мониторинга сетевой безопасности. При этом следует отметить, что область применения предлагаемой системы может выходить за рамки мониторинга безопасности сети IoT. За счет настройки правил корреляции событий эта система может достаточно эффективно решать различные задачи, связанные с обработкой Больших данных.

Рассматриваемая в статье система параллельной обработки данных обладает следующими особенностями. Во-первых, она благодаря использованию технологии Complex Event Processing (CEP) реализует в режиме «на лету» основные функции мониторинга, которыми являются нормализация, фильтрация, агрегация и корреляция данных. Во-вторых, кроме указанных выше функций, в этой системе реализована функция визуализации данных с использованием не только стандартных, но и специально разработанных моделей визуализации. В-третьих, эта система функционирует в условиях вычислительных ограничений, свойственных элементам информационной инфраструктуры сетей IoT. В силу наличия таких ограничений, слабые с точки зрения вычислительной способности устройства вынуждены передавать информацию на

верхние уровни сети для последующего анализа. При этом в качестве программно-инструментальной платформы для построения системы мониторинга используется открыто распространяемая программная среда Hadoop, которая в настоящее время является наиболее распространенной и достаточно гибкой платформой, позволяющей создавать системы параллельной обработки [5-7]. Эти особенности определяют теоретическую и практическую значимость настоящей работы.

Таким образом, цель данной статьи заключается в рассмотрении основных архитектурных и системных решений, позволяющих реализовать систему параллельной обработки данных, которая обладает указанными выше особенностями и может быть применима для мониторинга безопасности сетей IoT. Мониторинг безопасности способен выявлять нежелательную и вредоносную активность элементов сети IoT. Эта активность в дальнейшем может стать целью для задач обеспечения безопасности.

2. Состояние исследований. Технология CEP, используемая для разработки предлагаемой системы мониторинга безопасности для сетей IoT, в последнее время находится в фокусе внимания исследований, посвященных обработке Больших данных. Во многих работах проводится анализ и рассматриваются решения, связанные с реализацией технологии CEP в системах, основанных на Hadoop.

Система, которая рассматривается в [8], основана на Hadoop-совместимом программном обеспечении и платформе Java. Эта система имеет три компонента: компонент взаимодействия с пользователем, компонент анализа данных и компонент хранения данных. Взаимодействие между компонентами осуществляется с помощью запросов и управляется контроллерами Java Servlet. Однако эта система ориентирована только на обработку данных в веб-приложениях. Кроме того, отсутствие результатов экспериментальной оценки не позволяют утверждать о возможности ее применения для мониторинга безопасности сетей IoT.

Работа [9] рассматривает систему анализа Больших данных в медицинских информационных инфраструктурах. Выбор Hadoop был обоснован тем, что это средство наиболее популярно для целей обработки больших массивов данных. Система реализована на Java, принадлежит к открытому программному обеспечению под эгидой Apache и использует относительно простую программную модель. Архитектура предложенной системы включает адаптер событий (Event Adaptor), механизм CEP-анализа (CEP Analysis Engine) и генератор отчетов о событиях (Report & Event Generator). Система ориентирована на совместное использование с ERP-системами медицинских институтов. Механизм анализа содержит сборщик событий (Event Collector), ана-

лизатор данных (Data Analyzer) и сервер хранения данных (Storage Server). К сожалению, авторы этой работы не приводят результаты экспериментальной оценки рассматриваемой системы.

В [10] рассматривается CEP-система, предназначенная для обработки больших массивов данных при управлении движением городского пассажирского автотранспорта. Предложенная система объединяет два подхода: CEP и распределенную потоковую обработку (Distributed Stream Processing Systems — DSPS) [11]. CEP поддерживается системой Esper, подход DSPS — системой Storm. Система Hadoop играет роль интегратора, объединяющего эти два подхода. Кроме того, Hadoop обеспечивает анализ исторических данных. Экспериментальная оценка показала высокую масштабируемость рассматриваемой системы. Однако, по нашему мнению, применение ее для мониторинга IoT затруднено из-за больших требований к вычислительным ресурсам. В то же время достигнутые значения производительности этой системы будут служить ориентиром в нашей работе.

Во многих работах исследуются вопросы, связанные с совершенствованием технологии CEP. В работе [12] представлена CEP-система, в которой реализован высокоуровневый язык запросов к событиям (High-Level Event Query Language), близкий к SQL. Для этого языка разработаны алгоритмы оптимизации запросов. Критерием оптимизации является минимум времени работы процессора. В работе [13] представлен язык для типовых запросов в среде CEP, который позволяет более быстро обрабатывать сложные запросы. Однако широкое применение этих систем для мониторинга сетей IoT является затруднительным вследствие необходимости как изучения этих языков конечными пользователями, так и использования дополнительных программных средств, поддерживающих эти языки запросов.

В работе [14] рассматривается платформа, позволяющая интегрировать CEP и методы интеллектуального анализа данных (Data Mining). В качестве примера рассматривается сценарий работы «умного» города. По этой причине данная работа вызывает интерес как пример использования CEP в сетях IoT. Платформа содержит модуль интеграции и предобработки данных, в котором выполняется Data Mining. Однако, по нашему мнению, Data Mining на больших объемах требует значительных вычислительных затрат. Поэтому эти результаты в сетях IoT имеют ограниченное применение.

В работе [15] рассматривается CEP-система, предназначенная для сбора и предобработки данных с RFID. Результаты обработки потоков событий в этой системе хранятся в базе данных MySql. Однако, несмотря на то, что эта система является примером реализации техно-

логии CEP в сети IoT, она не может быть успешно использована для мониторинга безопасности сетей IoT, так как вопросы параллельной обработки больших данных в ней не рассматривались.

В работах [16, 17] предлагаются платформы, позволяющие обрабатывать веб-данные с помощью технологии CEP. В работе [16] веб-данные предварительно переводятся в события легковесной структуры. Работа [17] рассматривает систему, которая позволяет описывать и производить мониторинг сложных событий во времени, близком к реальному. Однако, несмотря на то, что оба предложенных подхода осуществимы, вопросы предварительной обработки событий на основе параллельных вычислений в этих работах не рассматривались.

В работе [18] предлагается подход, названный «активный» CEP, в котором поддерживается точность параллельного выполнения потоков через внедрение активной поддержки правил с помощью механизма CEP. Активные правила помогают поддерживать целостность CEP-транзакций, включая транзакции предобработки событий. Однако распространение этих результатов на параллельную обработку событий в IoT представляется преждевременным.

В работах [19, 20] рассматривается хорошо масштабируемая инфраструктура анализа потоков для CEP, которая для распараллеливания нагрузки осуществляет разбиение запроса на подзапросы и назначает каждому из подзапросов субкластер вычислительных средств. Авторы предложили и исследовали различные стратегии формирования подкластеров. Однако результаты, полученные в этих работах, ориентированы на кластерную вычислительную инфраструктуру с очень большим количеством узлов. По этой причине ее использование в сетях IoT является затруднительным.

Работа [21] предлагает использовать для балансировки нагрузки генетические алгоритмы, в которых учитываются характеристики каналов связи между узлами сети IoT (пропускная способность, входная нагрузка). Однако применение генетических алгоритмов при обработке больших данных может нарушить реальный масштаб времени.

Достаточно близкой к нашей можно считать работу [22], в которой представлена основанная на Hadoop платформа обработки Больших данных в интересах обеспечения общественной безопасности. Платформа содержит сервер приложений и оболочку экспертной системы. Она реализует получение данных с сенсоров, отображение их на интерактивной карте и выполнение определенных действий в соответствии с заданными правилами. Однако результаты экспериментальной оценки производительности этой системы в этой работе отсутствуют.

Интересные результаты по обработке больших данных представлены в работе [23]. Предлагаемая в ней платформа обеспечивает

обработку больших массивов документальных данных на основе парадигмы MapReduce и с использованием системы управления документальными базами данных Mongo. Авторы предложили различные реализованные на этой платформе способы агрегации больших документальных данных. Однако они не привели результаты экспериментальных исследований предлагаемой платформы.

Подводя итоги анализа состояния исследований в области систем параллельной обработки Больших данных и применения CEP для IoT, можно сделать следующие выводы. Во-первых, существуют работы, посвященные реализации CEP в сетях, подобных IoT. Однако в них вопросы параллельной обработки Больших данных не рассматриваются. Во-вторых, существуют CEP-системы, в которых присутствует параллельная обработка больших данных. Однако эти системы не могут быть реализованы в IoT из-за существующих ограничений по пропускной способности каналов связи и производительности вычислительных узлов. Наконец, мониторинг безопасности компьютерных сетей работы является сравнительно новой и недостаточно исследованной областью для систем параллельной обработки Больших данных. Этим подтверждается высокая актуальность нашей работы.

3. Архитектура системы параллельной обработки данных для мониторинга безопасности сетей IoT. В настоящем разделе рассмотрим общую архитектуру системы параллельной обработки данных для мониторинга безопасности сетей IoT и особенности функционирования ее компонентов.

Общая архитектура системы. Система параллельной обработки данных для мониторинга безопасности сетей IoT предназначена для сбора и предварительной обработки больших объемов информации о событиях безопасности, которые поступают в систему от конечных пользовательских устройств («вещей») и элементов сетевой инфраструктуры (маршрутизаторы, антивирусные средства, операционные системы, СУБД, межсетевые экраны и т.д.). Эти события хранятся в соответствующих журналах безопасности.

Источники событий безопасности генерируют большие объемы разнородного трафика, которые можно отнести к категории Больших данных. Поэтому система может конкурировать с традиционными системами сетевой безопасности в случае, если последние не способны справляться с растущими объемами трафика в сети IoT.

Система ориентирована на реализацию в среде Hadoop и включает следующие функциональные компоненты:

– компонент сбора данных, отвечающий за своевременное и достоверное поступление в систему информации о событиях безопасности от источников различных типов;

– компонент хранилища данных, обеспечивающий надежное хранение данных и оперативную обработку запросов;

– компонент нормализации и анализа данных, осуществляющий преобразование собираемых данных к единому формату и выполнение над ними основных операций предварительной обработки;

– компонент визуализации данных, позволяющий в реальном времени проводить визуальный анализ с помощью предварительно разработанных моделей визуализации.

Общая архитектура системы, включающая перечисленные выше компоненты и связи между ними, представлена на рисунке 1. Стрелками указаны направления потоков данных.

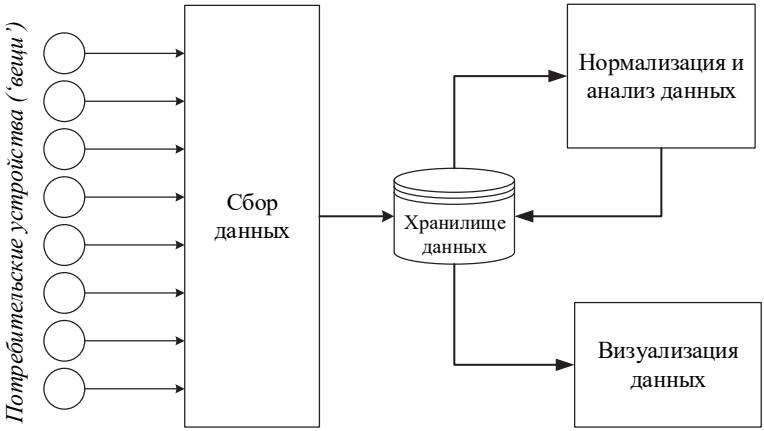


Рис. 1. Архитектура системы

Узлы контролируемой сети IoT отправляют данные для анализа компоненту сбора данных.

Компонент сбора данных организует распределенный прием данных и их хранение в компоненте хранилища данных.

Компонент нормализации и анализа данных получает данные для работы от компонента хранилища данных и в него же передает обратно на сохранение результата своей работы.

Компонент визуализации данных получает от компонента хранилища данных результирующие данные, которые были сформированы компонентом нормализации и анализа данных, и представляет их в виде, удобном для дальнейшего восприятия оператором.

Компонент сбора данных. Этот компонент предназначен для сбора трафика на машинах контролируемой сети IoT и его отправки в компонент хранилища данных.

Узлы (машины), участвующие в работе компонента сбора, по своему функциональному предназначению подразделяются на следующие типы: собирающие, предназначенные для сбора данных; координирующие, предназначенные для управления сбором данных; принимающие, предназначенные для приема собираемых данных. Взаимосвязь между этими типами машин показана на рисунке 2.

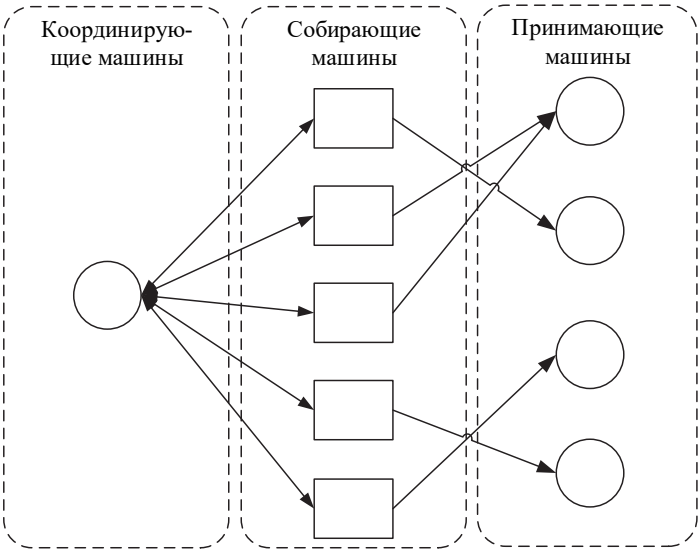


Рис. 2. Взаимосвязь между машинами в компоненте сбора данных

Как правило, координирующая машина присутствует в одном экземпляре и занимается балансировкой потоков трафика, идущего с собирающих машин на принимающие машины.

На собирающие машины контролируемой сети устанавливаются приложения-агенты, выполняющие следующие функции:

– подключение к координирующей машине;

– получение адреса принимающей машины для отправки логов (записей) трафика;

– получение списка принимающих машин для включения их во множество машин, не образующих трафик, который регистрируется при сборе;

– запуск сбора трафика;

– отправка логов трафика на принимающую машину.

Принимающие машины получают логи трафика от собирающих машин и переправляют их в компонент хранилища данных.

Компонент хранилища данных. Этот компонент поддерживает работу распределенного хранилища данных, которое представлено с помощью распределенной файловой системы Hadoop (Hadoop Distributed File System — HDFS) [24]. Файловая система HDFS делит большие файлы на блоки (по умолчанию их размер равен 64 МБ) и сохраняет их на узлах, называемых DataNode. Метаданные, показывающие, каким образом входные данные распределены по узлам DataNodes, хранятся в узле, называемом NameNode. Узел NameNode управляет метаданными в HDFS и администрирует запросы на выдачу данных пользователям. Каждый узел DataNode сохраняет данные, копируя их с компьютера пользователя и распространяя копии блоков между другими узлами DataNode.

DataNode-процессы запущены на каждой принимающей машине. NameNode-процесс запущен только на координирующей машине. Все логи входного трафика загружаются в систему HDFS и доступны после этого для дальнейшей обработки компонентом нормализации и анализа.

Компонент нормализации и анализа данных. Данные, сохраненные компонентом сбора, становятся доступными из системы HDFS для компонента нормализации и анализа.

Нормализация данных заключается в приведении всех входных данных к единому внутреннему формату. В качестве такового формата выбран формат CSV (Comma-Separated Values).

Процесс нормализации данных, реализованный в системе, имеет два режима работы: основной и дополнительный. В основном режиме просматривается содержимое входного файла. Для каждой встретившейся в нем записи формируется отдельная строка в выходном файле с жестко заданным набором атрибутов, определяющим структуру записи. В дополнительном режиме используется фильтр, позволяющий ограничивать выбор только тех записей, которые удовлетворяют описанным в фильтре условиям. Это позволяет снять излишнюю нагрузку, которая может быть вызвана ненужным просмотром данных, не участвующих в анализе.

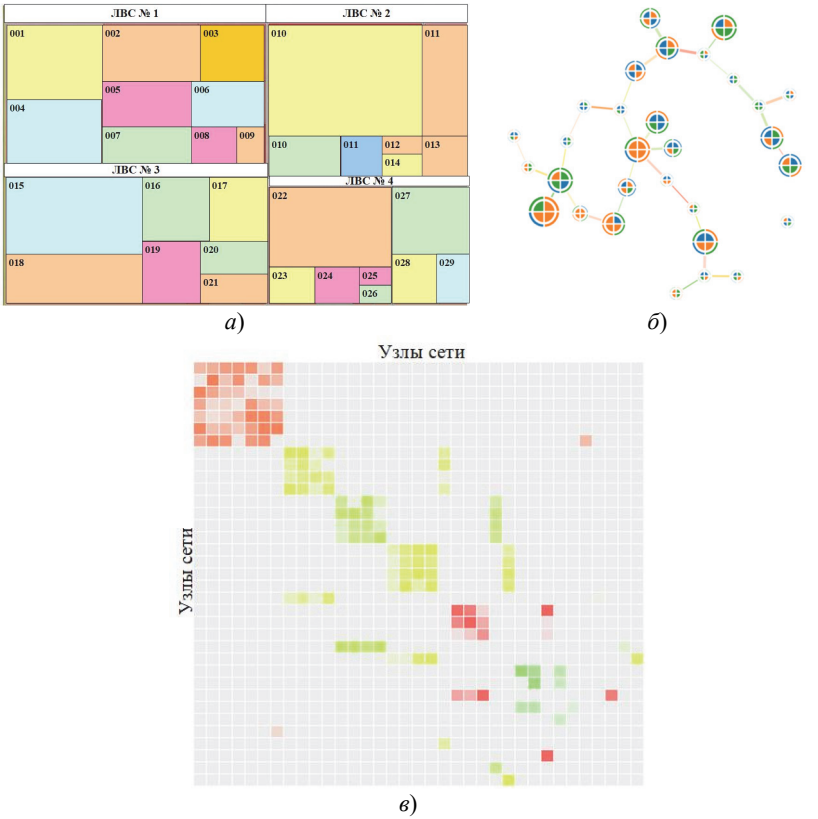
Кроме того, при нормализации с заранее заданной пользователем частотой (ротацией) генерируются выходные CSV-файлы. После этого осуществляется формирование очередных полей, которые записываются в созданные файлы.

Процесс анализа данных заключается в выполнении функций агрегации и корреляции данных. Агрегация представляет собой вычисление мер центральной тенденции (среднего значения, медианы, моды и квантилей). Корреляция призвана выявлять во входном потоке трафика аномальные события негативного характера, используя априори заданные правила. Результаты выполнения функций агрегации и корреляции данных представляются в формате CSV.

Компонент визуализации данных. Этот компонент предназначен для представления пользователю результирующих данных, полученных на этапе анализа в удобном для их визуальной оценки виде. Компонент берет исходные данные из системы HDFS, где они были сохранены после обработки компонентом нормализации и анализа.

Компонент визуализации использует различные стандартные и нестандартные модели визуализации. К стандартным моделям визуализации относятся: гистограммы; круговые диаграммы; линейные графики. Для реализации этих моделей визуализации используются встроенные возможности графических библиотек языка программирования Java.

К нестандартным моделям визуализации (рисунок 3), специально реализованным для мониторинга безопасности сетей IoT, относятся: карты деревьев; графы, дополненные глифами; матрицы.



а) *б*)

в)

Рис. 3. Виды нестандартных моделей визуализации: а) карта деревьев; б) граф с глифами; в) — матрица

Карта деревьев (рисунок 3*а*) отображает элементы сети IoT в виде прямоугольников. Индикаторами различных атрибутов сетевой безопасности являются: цвет, яркость и размер фигуры.

В графе с грифами (рисунок 3*б*) узлы соответствуют элементам сети IoT, а дуги — каналам связи.

Под глифом понимается сектор на круге, закрывающем узел графа. Уровень защищенности элемента характеризуется цветом и размером глифа.

Матрица (рисунок 3*в*) является моделью визуализации, отображающей уровень безопасности телекоммуникаций, соединяющих элементы сети IoT. Столбцы и строки матрицы соответствуют узлам, а ячейки соответствуют линиям связи, соединяющим эти узлы. Цвет и яркость ячейки позволяет кодировать любые две характеристики, например, «важность элемента» и «степень опасности атаки».

4. Реализация системы. С целью оценки производительности разработанной архитектуры был создан вычислительный кластер на базе Hadoop 2.6.4. Аппаратной платформой создания кластера послужила материнская плата Supermicro X9DRL-3F с двумя процессорами Intel Xeon E5-2620 v2 @ 2.1 GHz на борту. На базе гипервизора ESXi 6.0 было создано 7 (6 рабочих и одна главная) виртуальных машин с Ubuntu Server 14.04. Каждой машине было выделено 2 потока с общей тактовой частотой 2 GHz процессора. Также для каждой виртуальной машины было зарезервировано и ограничено 4 GB RAM.

Эти ограничения были вызваны необходимостью соответствия ограничениям типовой сети IoT. Традиционные компьютерные сети могут иметь более мощные вычислительные ресурсы для анализа их трафика. В частности, в традиционных сетях для этих целей можно использовать конфигурацию, в которой одна рабочая/главная машина будет соответствовать Supermicro X9DRL-3F (или аналогичной с одним гнездом под процессор) с процессором Intel Xeon E5-2620 v2 @ 2.1 GHz с 64 GB RAM под управлением Ubuntu Server.

Исходный код Hadoop был собран на одной виртуальной машине и скопирован на остальные машины.

Для моделирования сети IoT были использованы возможности проекта GRANIT, который разрабатывался в Российском прикладном центре компьютерных сетей (http://arccn.ru/media/1385?lang=en). Генерация структуры сети IoT производилась автоматически в соответствии со схемой, исполняемой во встроенном графическом редакторе. Топология сети показана на рисунке 4, где приведен фрагмент панели управления проекта GRANIT.

Первоначальный вариант структуры сети IoT содержал 20 собирающих машин (источников данных) и 10 принимающих машин. Ис-

точники были разбиты на 2 группы: Rack2 и Rack3, на которые было установлено ПО стенда генерации текстовых данных. В группу Rack2 входили машины Virt-slave-2-1, …, Virt-slave-2-10, с ПО, имитирующим кондиционер с функциями регулировки температуры, влажности и тому подобное. В группу Rack3 входили машины Virt-slave-3-1, …, Virt-slave-3-10 с ПО, имитирующим холодильник с функциями регулировки температуры, распознавания содержимого и так далее. Собирающие машины составляли группу Rack1 (машины Virt-slave-1-1, …, Virt-slave-1-10), они получали данные о состоянии устройств в Rack1 и Rack2. Каждая группа была связана с вычислительным кластером.

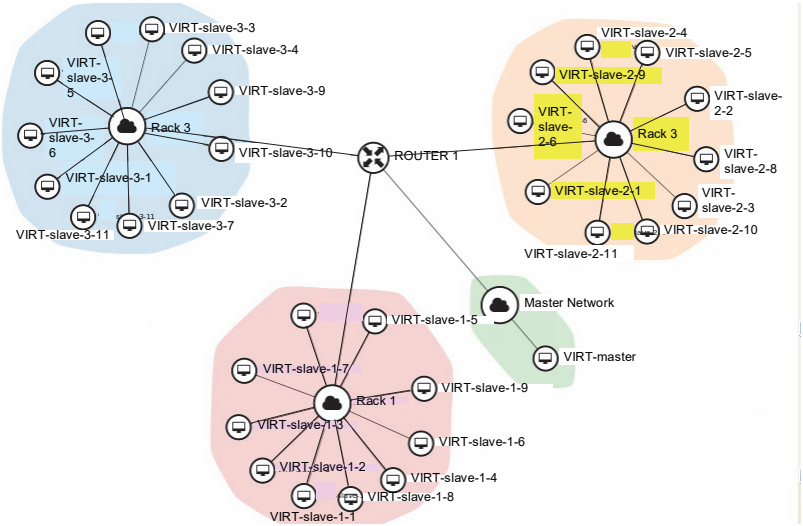


Рис. 4. Контрольная панель топологии сети GRANIT

Управление ресурсами вычислительного кластера осуществлялось с помощью программного средства YARN (Yet Another Resource Negotiator). Компонент YARN входит в состав стандартной конфигурации Hadoop. YARN выделяет ресурсы Hadoop для нужд запускаемых задач и выполняет функцию планировщика задач.

Средство YARN состоит из модуля ResourceManager (RM), который работает на главном узле, и модуля NodeManager (NM), который работает на рабочем узле. Когда на модуль RM поступает запрос на выполнение задачи, он выделяет ресурсы (создает контейнер) у одного из модулей NM и запускает на них процесс ApplicationMaster (AM). Этот процесс определяет, какие ресурсы требуются задаче для ее выполнения, и отправляет запрос модуля RM для выделения контейнеров у моделей NM.

Далее процесс AM получает информацию, на каких модулях NM выделены контейнеры для задачи, и запускает на них процессы выполнения задачи. К тому же процесс AM контролирует штатное исполнение задачи и перезапускает процессы, если они завершились преждевременно.

Модуль NodeManager контролирует состояние конкретных рабочих узлов. Он определяет доступные вычислительные ресурсы и отправляет их модулю RM. Кроме того, он создает и удаляет контейнеры по запросу модуля RM, завершает процессы, которые выходят за рамки ресурсов контейнера, и периодически отправляет тактовые сигналы модулю RM.

Для проведения экспериментов были использованы стандартные настройки YARN и MapReduce, приведенные в таблице 1.

Таблица 1. Настройки YARN и MapReduce

Компонент Hadoop	Параметр	Значение
YARN	yarn.nodemanager.resource.memory-mb	8096
	yarn.scheduler.minimum-allocation-mb	1024
	yarn.scheduler.maximum-allocation-mb	8096
	yarn.scheduler.minimum-allocation-vcores	1
	yarn.scheduler.maximum-allocation-vcores	32
	yarn.nodemanager.vmem-pmem-ratio	2.1
MapReduce	mapreduce.map.memory.mb	1024
	mapreduce.reduce.memory.mb	1024
	mapreduce.map.cpu.vcores	1
	mapreduce.reduce.cpu.vcores	1

Важной задачей реализации системы являлась подготовка исходных данных для моделирования. Входные потоки данных должны были содержать информацию о двух типах событий безопасности:

– о событиях, описывающих сетевой трафик и отражающих легитимную и вредоносную активность;

– о событиях, заключающихся в изменениях программно-аппаратных конфигураций персональных компьютеров, сетевого оборудования и потребительских устройств.

Для этой цели входные данные формировались двумя способами:

– программно-аппаратным стендом генерации тестовых наборов гетерогенных данных;

– на основе внешней базы экспериментальных данных о трафике компьютерной сети.

Программно-аппаратный стенд генерации тестовых наборов гетерогенных данных включал следующие элементы:

– десять персональных компьютеров, из них 8 были реализованы в рамках системы виртуализации;

– три маршрутизатора ASUS RT-N16, по одному для каждой подсети стенда;

– четыре элемента распределенной сети электронных потребительских устройств на базе контроллера Arduino Yun;

– один межсетевой экран, который осуществлял фильтрацию пакетов по заранее настроенным правилам.

В состав программных средств стенда входили следующие программные средства:

– сканер безопасности Nessus Home;

– система анализа трафика Wireshark 1.12.2;

– сетевой сканер NMap 6.47;

– система MetaSploit Framework 4.0;

– система обнаружения вторжений Snort 2.9.7.0.

В качестве внешней базы экспериментальных данных о трафике компьютерной сети использовалась база данных MAVILab [25]. Эта база данных содержит в архивном виде данные о реальном сетевом трафике, циркулирующем между Японией и США и содержащем пакеты различных форматов (tcp, http, ftp и т.д.). Данные из этого источника удовлетворяют требованиям гетерогенности, которые характерны и для сетей IoT. Непосредственно для анализа используются наборы данных, получаемые путем интеграции фонового трафика MAVILab и трафика, сгенерированного тестовым стендом IoT.

Файлы, входящие в состав внешней базы данных, соответствуют сетевому трафику длительностью до 15 минут. Все файлы имеют формат *.pcap.

Использование внешней базы данных позволило включить в состав тестовых данных события, описывающие реальный сетевой трафик, отражающий легитимную и вредоносную активность. Это сделало возможной проверку разработанной системы в условиях, максимально приближенных к условиям реальной эксплуатации системы.

5. Экспериментальные исследования. На текущем этапе создания системы обработки данных для мониторинга безопасности сетей IoT экспериментальной оценке и всестороннему тестированию был подвергнут только компонент нормализации и анализа данных. Цель экспериментов заключалась в определении производительности этого компонента при решении задач распределенного анализа собираемых данных в интересах мониторинга сетевой безопасности. При этом входные потоки представлялись в виде файла большого объема, поступающего на обработку в разработанный компонент. Задержки между событиями безопасности, имеющие место в реальных потоках, не

учитывались, так как при обработке больших объемов данных их влияние не является существенным. Учет таких задержек планируется осуществлять в дальнейших работах.

Анализ входных потоков данных, представляющих собой слияние потоков, генерируемых программно-аппаратным стендом, и потоков, содержащихся во внешней базе данных, проводился на основе заданных правил. В частности, в качестве одного из таких типовых правил использовалось следующее утверждение: «событие безопасности о сканировании портов регистрируется только в том случае, если с одного и того же IP-адреса на другой IP-адрес были высланы пакеты на более чем 10 портов, причем на каждый порт пришлось менее 5 пакетов».

Также в качестве тестовой задачи проводилась агрегация по одному из полей заголовка TCP-пакета, в роли которого выступало поле `src_ip`, определяющее IP-адрес источника.

Всего тестовая программа анализа содержала четыре задачи MapReduce, из которых две отводились на исполнение выше описанного правила и две исполняли функцию агрегации.

Решение задачи анализа по правилу осуществлялось в два этапа. На первом этапе на фазе Map из всего tcp-заголовка в качестве ключа выбирался триплет <`src_ip`, `dst_ip`, `dst_port`>. В качестве значения передавалась единица. Таким образом, на фазе Reduce путем сложения сгруппированных по ключу значений вычислялось количество пакетов, пришедших на конкретный порт конечного IP-адреса с другого IP-адреса. При записи результата ключ оставался без изменения, а в качестве значения передавалось суммарное количество пакетов, поступающее на этот порт. Причем, согласно условию задачи, в выходной файл попадали только те пары «ключ — значение», где сумма пакетов, поступающих на порт, была меньше пяти.

На втором этапе анализа по приведенному выше правилу на фазе Map в качестве ключа выбиралась пара <`src_ip`, `dst_ip`>. В качестве значения отправлялась функция `count`(), определяющая количество пакетов, поступивших на порт (из предыдущего этапа MapReduce). На фазе Reduce количество значений по каждому ключу оказывалось равным количеству портов, на которые передавались пакеты с исходящего IP-адреса на целевой IP-адрес. Согласно условию задачи, если количество портов больше 10, то ключ и количество портов записывались в выходной файл.

Агрегация проводилась по одному из полей tcp-заголовка. Эта задача решалась также в два этапа. На первом этапе, например, для поля `src_ip`, в фазе Map ключом являлось `src_ip`, а значением —

единица. Таким образом, в фазе Reduce путем сложения значений по ключам определялось, сколько вхождений имеет тот или иной `src_ip` в исходных логах. На втором этапе в фазе Map пришедшая пара «ключ — значение» передавалось на Reduce в качестве значения, а сам ключ оставался пустым. В фазе Reduce выполнялся поиск максимального и минимального числа вхождений `src_ip`, а также вычислялось среднее число вхождений.

В ходе экспериментальных исследований проводились многократные замеры времени обработки больших массивов данных, соответствующих входным потокам системы мониторинга сетевой безопасности, в зависимости от объемов этих файлов и количества рабочих машин, входящих в кластер Hadoop. Объемы файлов со входными потоками изменялись в диапазоне от 0,5 до 3 ГБ. Количество машина в вычислительном кластере равнялось 3, 5 или 7 единиц.

Результаты экспериментальной оценки времени обработки файлов входных потоков в зависимости от объема нагрузки и количества узлов в кластере представлены в таблице 2 и на рисунке 5. Учитывалось «чистое» время обработки нагрузки без учета настройки Hadoop. Экспериментальные оценки рассчитывались как средние арифметические значения за 50 прогонов.

Таблица 2. Длительности обработки входной нагрузки (сек)

Объем входной нагрузки, ГБ	Количество узлов в кластере		
	3	*5*	*7*
0,5	76,41	61,85	51,65
1,0	135,98	106,38	102,00
1,5	208,18	161,35	158,93
2,0	322,28	202,32	179,05
2,5	409,81	259,08	210,91
3,0	522,30	357,98	245,12

Так как рассматриваемая система мониторинга сетевой безопасности является разновидностью CEP-систем, то представляет интерес зависимость времени обработки данных не от объема входной нагрузки, а от количества содержащихся в ней событий. Оценка объема данных, содержащихся в файлах формата *.pcap, показывает, что средний объем одной записи в файле формата *.pcap составляет примерно 600…620 байт. Выберем для него значение 612 байт.

Тогда можно перейти от данных, представленных в таблице 2 и на рисунке 5, к зависимости времени обработки данных от количества узлов в кластере и от количества событий, поступивших на обработку (рисунок 6).

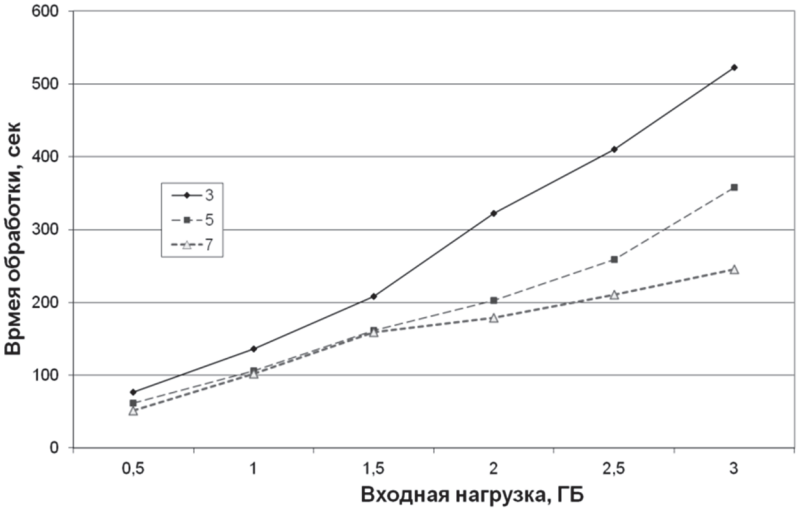


Рис. 5. Зависимость длительности обработки данных от объема входной нагрузки и количества узлов в кластере

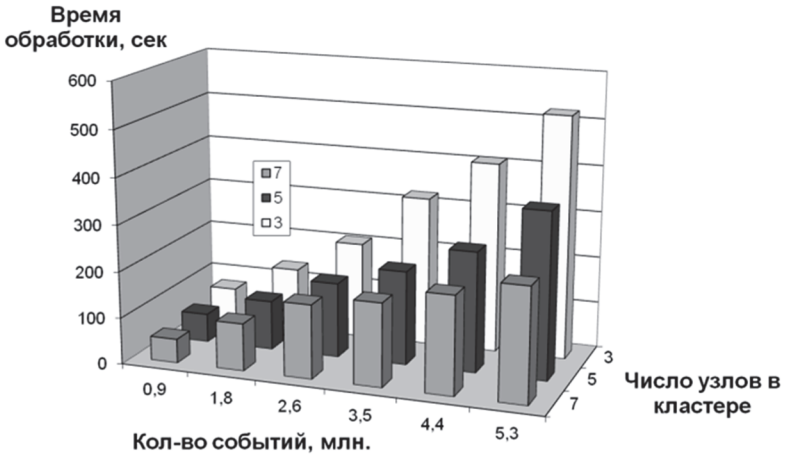


Рис. 6. Зависимость длительности обработки данных от количества обрабатываемых событий и количества узлов в кластере

От зависимости, приведенной на рисунке 6, перейдем к оценке производительности системы мониторинга, которую будем измерять с помощью количества событий, обработанных за единицу времени. Вве-

дем в рассмотрение интегральную и частную производительности системы. Интегральная производительность системы определяется как:

$$ИнтегральнаяПроизводительность = \frac{ЧислоСобытий}{ВремяОбработки}, \quad (1)$$

где *ЧислоСобытий* — общее количество обработанных событий; *ВремяОбработки* — среднее время обработки событий (сек).

Частная производительность системы определяется следующим образом:

$$ЧастнаяПроизводительность = \frac{ЧислоСобытий}{ВремяОбработки * ЧислоУзлов}, \quad (2)$$

где *ЧислоУзлов* — количество узлов в вычислительном кластере.

Результаты расчета интегральной и частной производительности по формулам (1) и (2) приведены на рисунках 7 и 8.

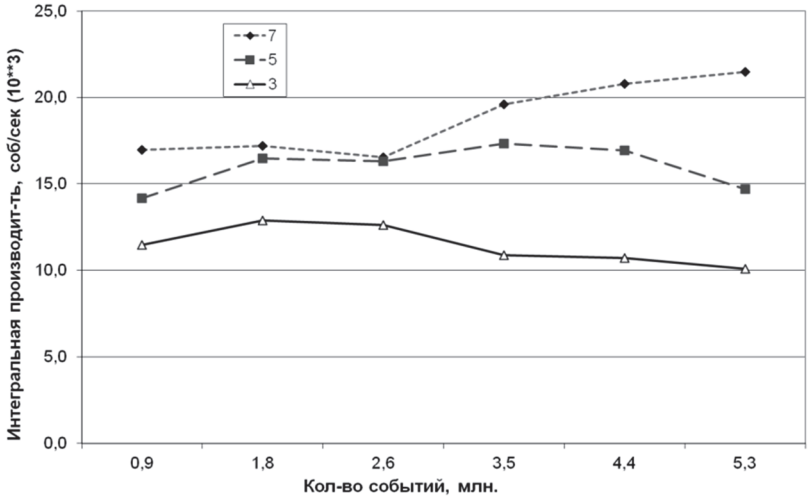


Рис. 7. Зависимость интегральной производительности системы от количества обрабатываемых событий и количества узлов в кластере

Анализ этих данных позволяет сделать следующие выводы. Во-первых, увеличение количества узлов в вычислительном кластере приводит к увеличению интегральной производительности системы. Это видно, если сравнивать между собой графики на рисунке 7 при различных значениях параметра *ЧислоСобытий*.

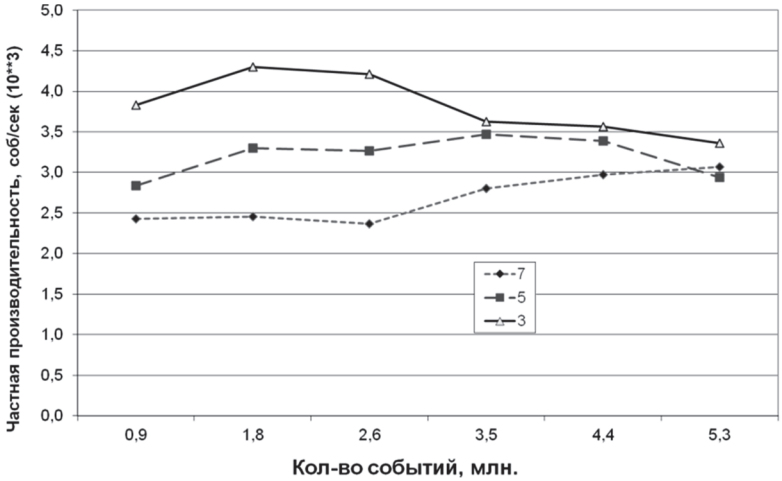


Рис. 8. Зависимость частной производительности системы от количества обрабатываемых событий и количества узлов в кластере

В то же время для частной производительности системы наблюдается другая картина. Частная производительность системы при конфигурации, в которой *ЧислоУзлов* = 3, практически при всех объемах входной нагрузки является максимальной. Правда, при большой и средней нагрузке она начинает падать. В этом случае она практически совпадает с частной производительностью для конфигурации, в которой *ЧислоУзлов* = 5. А при большой нагрузке для всех трех конфигураций частная производительность совпадает.

Кроме того, из рисунка 7 видно, при каких нагрузках и конфигурациях частная производительность достигает своего максимума.

Для условия *ЧислоУзлов* = 3 максимум частной производительности достигается при малой нагрузке, находящейся в диапазоне от 0,9 до 2,6 миллионов событий. В этом случае частная производительность достигает пика, равного $4{,}3*10^3$ событий в секунду. Для конфигурации, в которой *ЧислоУзлов* = 5, максимум частной производительности достигается при средней нагрузке (от 2,6 до 4,4 миллионов событий). Этот максимум равняется $3{,}5*10^3$ событий в секунду. Наконец, для условия *ЧислоУзлов* = 7 максимум частной производительности достигается при большой нагрузке (от 4,4 до 5,6 миллионов событий). Тенденция увеличения частной производительности сохраняется. Можно ожидать, что при дальнейшем увеличении входной нагрузки значение частной производительности будет и дальше увеличиваться.

Такое поведение частной производительности при различной входной нагрузке объясняется следующими причинами. Во-первых,

для конфигурации, в которой *ЧислоУзлов* = 3, каждый узел кластера очень быстро нагружается. Однако при дальнейшем увеличении нагрузки кластерные узлы испытывают перегрузку, что отражается на снижении частной производительности. Насыщение узлов при условии *ЧислоУзлов* = 5 происходит при средней входной нагрузке. Дальнейшее ее увеличение может привести в перегрузке узлов кластера. При условии *ЧислоУзлов* = 7 для рассматриваемых объемов входной нагрузки никакой перегрузки узлов не наблюдается. Узлы постепенно входят в насыщение.

Некоторые расхождения измерений на рисунках 7 и 8 от описанных выше общих тенденций поведения интегральной и частной производительности можно объяснить рядом причин, которые определяются принципами работы Hadoop и, в частности, файловой системы HDFS. В наших экспериментах система HDFS была развернута на одних и тех же жестких дисках. В результате в ряде случаев работу Hadoop могли тормозить блокировки при обращении к узлам DataNode, расположенных на одном и том же жестком диске. Нам следовало бы разнести все узлы DataNode по разным жестким дискам. Однако этот случай мы рассматриваем как будущие исследования.

Завершая анализ полученных экспериментальных результатов, следует отметить, что они не противоречат результатам, полученным в других известных работах [9, 10, 12]. Это позволяет говорить, что производительность разработанной нами системы является не хуже, чем для известных систем, несмотря на наличие вычислительных ограничений, свойственных сетям IoT.

6. Заключение. В настоящей статье предложен новый подход к построению систем мониторинга безопасности сетей IoT, основанный на реализации параллельной потоковой обработки данных о событиях безопасности. Согласно этому подходу, система мониторинга сетевой безопасности реализуется на платформе Hadoop с возможностью использования технологии CEP. Предлагаемая архитектура разрабатываемой системы параллельной обработки данных для мониторинга безопасности сетей IoT включает компоненты, ответственные за сбор, хранение, нормализацию и анализ, а также визуализацию данных. Нормализация, анализ и визуализация данных осуществляются в режиме «на лету». Хранение данных осуществляется в распределенной файловой системе Hadoop, что повышает надежность хранения и оперативность обработки запросов к данным.

Реализация системы для проведения ее экспериментальной оценки была выполнена с учетом вычислительных ограничений, свойственных сетям IoT. Входные потоки моделировались путем использо-

вания специального стенда, генерирующего события безопасности во фрагменте сети IoT, и использования внешней базы данных о трафике в реальной компьютерной сети. Экспериментальная оценка разработанной системы показала, что, несмотря на наличие ограничений в вычислительных ресурсах, система обладает достаточно высокой производительностью, сравнимой, а в некоторых случаях значительно превышающей известные реализации.

Дальнейшие исследования планируется проводить в направлении совершенствования разработанной системы мониторинга сетевой безопасности за счет повышения скорости обращений к блокам данных файловой системы HDFS и более широкого внедрения средств реализации CEP. В частности, планируется реализовать компонент нормализации и анализа данных в среде Spark. Также планируется изучить влияние пропускной способности устройств сети IoT на эффектность работы системы.

Литература

1. *Котенко И.В., Саенко И.Б.* SIEM-системы для управления информацией и событиями безопасности // «Защита информации. Инсайд». 2012. № 5. С. 54–65.
2. *Котенко И.В., Саенко И.Б.* Построение системы интеллектуальных сервисов для защиты информации в условиях кибернетического противоборства // Труды СПИИРАН. 2012. № 3(22). С. 84–100.
3. *Котенко И.В.* Интеллектуальные механизмы управления кибербезопасностью // Труды Института системного анализа Российской академии наук. 2009. Т. 41. С. 74–103.
4. *Kotenko I., Chechulin A.* Attack Modeling and Security Evaluation in SIEM Systems // International Transactions on Systems Science and Applications. 2012. vol. 8. pp. 129–147.
5. *De Carvalho O.M., Roloff E., Navaux O.A.* A Survey of the State-of-the-art in Event Processing // Proceedings of the 11th Workshop on Parallel and Distributed Processing (WSPPD). 2013. 16 p.
6. Apache Hadoop 3.0.0-alpha4. URL: http://hadoop.apache.org/docs/current/ (дата обращения: 01.06.2017).
7. Holmes A. Hadoop in Practice // Manning Publications Co. 2012. 536 p.
8. *Scherbakov M. et al.* A Design of Web Application for Coomplex Event Processing Based on Hadoop and Java Servlets // International Journal of Soft Computing. 2015. vol. 10. no. 3. pp. 218–219.
9. *Kim M.-J., Yu Y.-S.* Development of Real-time Big Data Analysis System and a Case Study on the Application of Information in a Medical Institution // International Journal of Software Engineering and Its Applications. 2015. vol. 9. no. 7. pp. 93–102.
10. *Zygouras N. et al.* Insights on a Scalable and Dynamic Traffic Management System // Proceedings of the 18th International Conference on Extending Database Technology (EDBT-2015). 2015. pp. 653–664.
11. *Cherniack M. et al.* Scalable Distributed Stream Processing // Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR'03). 2003. vol. 3. pp. 257–268.
12. *Schultz-Møller N.P., Migliavacca M., Pietzuch P.* Distributed Complex Event Processing with Query Rewriting // Proceedings of the Third ACM International Conference on Distributed Event-Based Systems. 2009. Article no. 4. 12 p.

13. *Zhang H., Diao Y., Immerman N.* On Optimization of Expensive Queries in Complex Event Processing // Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14). 2014. pp. 217–228.
14. *Moraru A., Mladenić D.* Complex Event Processing and Data Mining for Smart Cities // Proceedings of the 15th International Multiconference on Information Society (IS-2012). 2012. 4 p. URL: http://ailab.ijs.si/dunja/SiKDD2012/Papers/Moraru_CEP.pdf (дата обращения: 01.06.2017).
15. *Gyllstrom D. et al.* SASE: Complex Event Processing over Streams // 2006. arXiv preprint cs/0612128. 4 p. URL: https://arxiv.org/ftp/cs/papers/0612/0612128.pdf (дата обращения: 01.06.2017).
16. *Liu D., Pedrinaci C., Domingue J.* A framework for feeding Linked Data to Complex Event Processing engines // 1st International Workshop on Consuming Linked Data (COLD 2010) at The 9th International Semantic Web Conference (ISWC 2010). 2010. 12 p. URL: http://oro.open.ac.uk/26057/1/LiuEtAl_COLD2010.pdf (дата обращения: 01.06.2017).
17. *Anicic D., Rudolph S., Fodor P., Stojanovic N.* Stream reasoning and complex event processing in ETALIS // Semantic Web. 2012. vol. 3. no. 4. pp. 397–407.
18. *Wang D., Rundensteiner E.A., Ellison R.T.* Active Complex Event Processing over Event Streams // Proceedings of the VLDB Endowment. 2011. vol. 4. no. 10. pp. 634–645.
19. *Gulisano V., Jimenez-Peris R., Patino-Martinez M., Valduriez P.* StreamCloud: A Large Scale Data Streaming System // Proceedings of the 2010 International Conference on Distributed Computing Systems. 2010. pp. 126–137.
20. *Gulisano V. et al.* StreamCloud: An Elastic and Scalable Data Streaming System // IEEE Transactions on Parallel and Distributed Systems. 2012. vol. 23. no. 12. pp. 2351–2365.
21. *Kotenko I., Saenko I.* An Approach to Aggregation of Security Events in Internet-of-Things Networks Based on Genetic Optimization // Proceedings of the 16th IEEE International Conference on Scalable Computing and Communications (ScalCom 2016). 2016. pp. 657–664.
22. *Толстолес А.А. и др.* Платформа для разработки приложений Интернета вещей на основе модуля анализа больших данных // Информатика и кибернетика (ComCon-2016): Сб. научн. конф. Института компьютерных наук и технологий (ИКНТ). 2016. С. 192–194.
23. *Гончарова М.Н.* Механизм параллельной обработки больших объемов информации в документо-ориентированных системах управления базами данных // Вестник магистратуры. 2014. № 6–1(33). С. 52–56.
24. *Dwivedi K., Dubey S.K.* Analytical review on Hadoop Distributed file system // Proceedings of the 5th International Conference on Confluence the Next Generation Information Technology Summit (confluence). 2014. pp. 174–181.
25. MAVILab URL: http://www.fukuda-lab.org/mawilab/index.html (дата обращения: 01.06.2017).

Котенко Игорь Витальевич — д-р техн. наук, профессор, заведующий лабораторией проблем компьютерной безопасности, Федеральное государственное бюджетное учреждение науки Санкт-Петербургский институт информатики и автоматизации Российской академии наук (СПИИРАН). Область научных интересов: безопасность компьютерных сетей, в том числе управление политиками безопасности, разграничение доступа, аутентификация, анализ защищенности, обнаружение компьютерных атак, межсетевые экраны, защита от вирусов и сетевых червей, анализ и верификация протоколов безопасности и систем защиты информации, защита программного обеспечения от взлома и управление цифровыми правами, технологии моделирования и визуализации для противодействия кибер–терроризму. Число научных публикаций — 500.

ivkote@comsec.spb.ru, http://www.comsec.spb.ru; 14-я линия В.О., 39, Санкт-Петербург, 199178; р.т.: +7-(812)328-7181, Факс: +7(812)328–4450.

Саенко Игорь Борисович — д-р техн. наук, профессор, ведущий научный сотрудник лаборатории проблем компьютерной безопасности, Федеральное государственное бюджетное учреждение науки Санкт-Петербургский институт информатики и автоматизации Российской академии наук (СПИИРАН). Область научных интересов: автоматизированные информационные системы, информационная безопасность, обработка и передача данных по каналам связи, теория моделирования и математическая статистика, теория информации. Число научных публикаций — 350. ibsaen@comsec.spb.ru, http://www.comsec.spb.ru; 14-я линия В.О., 39, Санкт-Петербург, 199178; р.т.: +7(812)328-7181, Факс: +7(812)328-4450.

Кушнеревич Алексей Геннадьевич — младший научный сотрудник лаборатории проблем компьютерной безопасности, Федеральное государственное бюджетное учреждение науки Санкт-Петербургский институт информатики и автоматизации Российской академии наук (СПИИРАН). Область научных интересов: большие данные, анализ данных. Число научных публикаций — 4. kushnerevich@comsec.spb.ru, http://www.comsec.spb.ru; 14-я линия В.О., 39, Санкт-Петербург, 199178; р.т.: +7(812)328-7181, Факс: +7(812)328-4450.

Поддержка исследований. Работа выполнена при финансовой поддержке РФФИ (проекты 16-29-09482, 18-07-01369 и 18-07-01488), при частичной поддержке бюджетной темы № АААА-А16-116033110102-5, а также при государственной финансовой поддержке ведущих университетов Российской Федерации (субсидия 074-U01).

DOI 10.15622/sp.59.1

I.V. KOTENKO, I.B. SAENKO, A.G. KUSHNEREVICH

ARCHITECTURE OF THE PARALLEL BIG DATA PROCESSING SYSTEM FOR SECURITY MONITORING OF INTERNET OF THINGS NETWORKS

Kotenko I.V., Saenko I.B., Kushnerevich A.G. **Architecture of the Parallel Big Data Processing System for Security Monitoring of Internet of Things Networks.**

Abstract. Internet-of-Things networks are applied in many areas of people life now. A cornerstone in a issue of a possibility of further distribution and use of these networks is the aspect of security support. However, the features of these networks complicate the use of traditional means and systems of computer protection in them. One of such features is the need to analyze very large volumes of data, heterogeneous by the nature, in real time and with the minimum computing expenses. Taking into account the features of computational capabilities of Internet-of-Things networks the architecture of the system for parallel big data processing based on the data processing technology named as Complex Event Processing and the parallel computing platform Hadoop is offered. The issues directly connected to the architecture of the system and with implementation of its principal components are considered. These components are: data collection component, data storage component, data normalization and analysis component, and data visualization component. An interconnection between components is provided by means of the Hadoop Distributed File System that is a basis for creation of the distributed data storage. The data collection component organizes the distributed data acquisition and their storage in the data storage component. The data normalization and analysis component transforms data to a uniform format and processes them by means of correlation rules. The data visualization component presents data in a graphical form more suitable for further perception by the operator. The results of the experimental evaluation of the system performance confirming a conclusion about its high performance are discussed.

Keywords: Internet of things; security monitoring; Big Data; Complex Event Processing; Hadoop.

Kotenko Igor Vitalievich — Ph.D., Dr. Sci., professor, head of computer security problems laboratory, St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences (SPIIRAS). Research interests: computer network security, including security policy management, access control, authentication, network security analysis, intrusion detection, firewalls, deception systems, malware protection, verification of security systems, digital right management, modeling, simulation and visualization technologies for counteraction to cyber terrorism. The number of publications — 500. ivkote@comsec.spb.ru, http://www.comsec.spb.ru; 39, 14-th Line V.O., St. Petersburg, 199178, Russia; office phone: +7-(812)328-7181, Fax: +7(812)328–4450.

Saenko Igor Borisovich — Ph.D., Dr. Sci., professor, leading researcher of computer security problems laboratory, St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences (SPIIRAS). Research interests: automated information systems, information security, processing and transfer of data on data links, theory of modeling and mathematical statistics, information theory. The number of publications — 350. ibsaen@comsec.spb.ru, http://www.comsec.spb.ru; 39, 14-th Line V.O., St. Petersburg, 199178, Russia; office phone: +7(812)328-7181, Fax: +7(812)328-4450.

Kushnerevich Alexey Gennadievich — junior researcher computer security problems laboratory, St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences (SPIIRAS). Research interests: Big Data, data analysis. The number of publications — 4. kushnerevich@comsec.spb.ru, http://www.comsec.spb.ru; 39, 14-th Line V.O., St. Petersburg, 199178, Russia; office phone: +7(812)328-7181, Fax: +7(812)328-4450.

Acknowledgements. This research was supported by grants of RFBR (projects No. 16-29-09482, 18-07-01369 and 18-07-01488), by the budget (the project No. AAAA-A16-116033110102-5), and by Government of the Russian Federation (grant 074-U01).

References

1. Kotenko I.V., Saenko I.B. SIEM-systems for security information and event management. *Zashhita informacii. Insajd — Zaŝita informacii. Inside*. 2012. vol. 5. pp. 54–65. (In Russ.).
2. Kotenko I.V., Saenko I.B. Developing the system of intelligent services to protect information in cyber warfare. *Trudy SPIIRAN – SPIIRAS Proceedings*. 2012. vol. 3(22). pp. 84–100. (In Russ.).
3. Kotenko I.V. Intelligent mechanisms of cybersecurity management. *Trudy Instituta sistemnogo analiza Rossijskoj akademii nauk – Proceedings of the Institute of System Analysis of the Russian Academy of Sciences*. 2009. vol. 41. pp. 74–103. (In Russ.).
4. Kotenko I., Chechulin A. Attack Modeling and Security Evaluation in SIEM Systems. *International Transactions on Systems Science and Applications*. 2012. vol. 8. pp. 129–147.
5. De Carvalho O.M., Roloff E., Navaux O.A. A Survey of the State-of-the-art in Event Processing. Proceedings of the 11th Workshop on Parallel and Distributed Processing (WSPPD). 2013. 16 p.
6. Apache Hadoop 3.0.0-alpha4. Available at: http://hadoop.apache.org/docs/current/ (accessed: 01.06.2017).
7. Holmes A. Hadoop in Practice. Manning Publications Co. 2012. 536 p.
8. Scherbakov M. et al. A Design of Web Application for Coomplex Event Processing Based on Hadoop and Java Servlets. *International Journal of Soft Computing*. 2015. vol. 10. no. 3. pp. 218–219.
9. Kim M.-J., Yu Y.-S. Development of Real-time Big Data Analysis System and a Case Study on the Application of Information in a Medical Institution. *International Journal of Software Engineering and Its Applications*. 2015. vol. 9. no. 7. pp. 93–102.
10. Zygouras N. et al. Insights on a Scalable and Dynamic Traffic Management System. Proceedings of the 18th International Conference on Extending Database Technology (EDBT-2015). 2015. pp. 653–664.
11. Cherniack M. et al. Scalable Distributed Stream Processing. Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR'03). 2003. vol. 3. pp. 257–268.
12. Schultz-Møller N.P., Migliavacca M., Pietzuch P. Distributed Complex Event Processing with Query Rewriting. Proceedings of the Third ACM International Conference on Distributed Event-Based Systems. 2009. Article no. 4. 12 p.
13. Zhang H., Diao Y., Immerman N. On Optimization of Expensive Queries in Complex Event Processing. Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14). 2014. pp. 217–228.
14. Moraru A., Mladenić D. Complex Event Processing and Data Mining for Smart Cities. Proceedings of the 15th International Multiconference on Information Society15th International Multiconference on Information Society (IS-2012). 2012. 4 p. Available at: http://ailab.ijs.si/dunja/SiKDD2012/Papers/Moraru_CEP.pdf (accessed: 01.06.2017).

15. Gyllstrom D. et al. SASE: Complex Event Processing over Streams. 2006. arXiv preprint cs/0612128. 4 p. Available at: https://arxiv.org/ftp/cs/papers/0612/0612128.pdf (accessed: 01.06.2017).
16. Liu D., Pedrinaci C., Domingue J. A framework for feeding Linked Data to Complex Event Processing engines. Proceedings of the 1st International Workshop on Consuming Linked Data (COLD 2010) at The 9th International Semantic Web Conference (ISWC 2010). 2010. 12 p. Available at: http://oro.open.ac.uk/26057/1/LiuEtAl_COLD2010.pdf (accessed: 01.06.2017).
17. Anicic D., Rudolph S., Fodor P., Stojanovic N. Stream reasoning and complex event processing in ETALIS. *Semantic Web*. 2012. vol. 3. no. 4. pp. 397–407.
18. Wang D., Rundensteiner E.A., Ellison R.T. Active Complex Event Processing over Event Streams. Proceedings of the VLDB Endowment. 2011. vol. 4. no. 10. pp. 634–645.
19. Gulisano V., Jimenez-Peris R., Patino-Martinez M., Valduriez P. StreamCloud: A Large Scale Data Streaming System. Proceedings of the 2010 International Conference on Distributed Computing Systems. 2010. pp. 126–137.
20. Gulisano V. et al. StreamCloud: An Elastic and Scalable Data Streaming System. *IEEE Transactions on Parallel and Distributed Systems*. 2012. vol. 23. no. 12. pp. 2351–2365.
21. Kotenko I., Saenko I. An Approach to Aggregation of Security Events in Internet-of-Things Networks Based on Genetic Optimization. Proceedings of the 16th IEEE International Conference on Scalable Computing and Communications (ScalCom 2016). 2016. pp. 657–664.
22. Tolstoles A.A. et al. [Software platform based on Big-Data anasysis module for development Internet-of-Things applications]. *Informatika i kibernetika (ComCon-2016): Sb. nauchn. konf. Instituta komp'juternyh nauk i tehnologij (IKNT)* [Informatics and Cybernetics (ComCon-2016): Proceedings]. 2016. pp. 192–194. (In Russ.).
23. Goncharova M.N. [Mechanism of parallel Big Data analysis in document-oriented DBMSs]. *Vestnik magistratury — Gerald of magistracy*. 2014. vol. 6–1(33). pp. 52–56. (In Russ.).
24. Dwivedi K., Dubey S.K. Analytical review on Hadoop Distributed file system. Proceedings of the 5th International Conference on Confluence the Next Generation Information Technology Summit (confluence). 2014. pp. 174–181.
25. MAVILab Available at: http://www.fukuda-lab.org/mawilab/index.html (accessed: 01.06.2017).