Министерство науки и высшего образования РФ

Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И СУПЕРКОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

ТРУДЫ XX МЕЖДУНАРОДНОЙ КОНФЕРЕНЦИИ

Нижний Новгород, 23–27 ноября 2020 г.

Нижний Новгород
Издательство Нижегородского госуниверситета
2020

УДК 004.942+519.876.5
ББК 22.181я43
М34

М34 **Математическое моделирование и суперкомпьютерные технологии.** Труды XX Международной конференции (Н. Новгород, 23–27 ноября 2020 г.) / Под ред. проф. В.П. Гергеля. – Нижний Новгород: Изд-во Нижегородского госуниверситета, 2020. – 438 с.

Отв. за выпуск К.А. Баркалов

ISBN 978-5-91326-621-7

Сборник материалов Двадцатой Международной конференции «Математическое моделирование и суперкомпьютерные технологии», состоявшейся 23−27 ноября 2020 г. на базе Нижегородского государственного университета им. Н.И. Лобачевского, содержит тезисы докладов и короткие статьи, посвященные математическому моделированию сложных процессов и явлений, численным методам их исследования, а также проблемам разработки методов суперкомпьютерных вычислений для решения актуальных задач в различных областях науки, промышленности и образования.

Подробную информацию о конференции можно найти в сети Интернет по адресу http://agora.guru.ru/hpc2020

Поддержка конференции

Корпорация Intel

Компания Huawei

Группа компаний РСК

Научно-образовательный математический центр
«Математика технологий будущего»

ISBN 978-5-91326-621-7
ББК 22.181я43

HIGHLY LOADED EVENT DETECTION SYSTEM ARCHITECTURE

A.B. Mussina[1], S.S. Aubakirov[1], P. Trigo[2]

[1]Al-Farabi Kazakh National University, Almaty, Kazakhstan,
[2]GulAA; ISEL - Instituto Superior de Engenharia de Lisboa; BioISI - Biosystems & Integrative Sciences Institute / Agent and Systems Modeling, University of Lisbon, Portugal

Social networks already play a significant role in human's daily life. Therefore social networks have become an arena of enormous opportunities to perform data analysis. Social media analytics applies to digital marketing, social opinion analysis, political situation monitoring, natural disaster notification. It is possible to detect events on which people are reacting in every moment. Event detection is a powerful data analysing process useful for different areas. We want to construct social models and event patterns prediction model based on online social media event detection in real-time. In this part of the research work we will present highly loaded, fault-tolerant, scalable system for social media data collection and real-time analysis.

Key words: event detection, highly loaded, fault-tolerant, scalable architecture, Telegram.

1. Introduction

Social media is mostly free, powerful and highly-spreaded platform for brand communication with their audience. Company may get quick feedback and make influence analysis of its posts [1]. Different Online Social Networks (OSN) construct own relationships between consumers and producers [2]. In our main research we want to analyse social behaviour in the context of messaging and posts reaction within Telegram messenger. As of April 2020 the Kazakhstan's audience of Telegram reached 400 000 users, while at the beginning of 2019 it was approximately 200 000 users [3]. Telegram is relatively young OSN and has not yet been a subject of research. In April 2020 its monthly active users all over the world reached 400 million users. The authors of Telegram OSN provide many public tools for developers to build their own client application [4].

Event detection process has been applied for various OSNs and purposes. An "event" has different definitions according to context and application. First we consider the definition as "An occurrence causing change in the volume of text data that discusses the associated topic at a specific time. This occurrence is characterized by topic and time, and often associated with entities such as people and location" [5]. Event detection in OSN is important for social analysis, because it allows to estimate public interest in an occurred event. Moreover, events analysis lead to the detection of substantial sub-events [6]. For example, at the beginning of the 2020 year the whole world faced new virus COVID-19. Such high-level event subsumes many significant low-level sub-events such as illegal reselling essential items, bullying activity, misinformation spreading, growth of free online services and infection of celebrities.

Our goal is focused on the intellectual social modelling and event patterns prediction based on online social media in real-time. At the moment in our research we have analysed related work and implemented highly loaded, fault-tolerant, scalable system for social media data collection.

2. Related works

Event detection is fairly popular theme for researches. A comprehensive survey on event detection [5] describes four challenges in the event detection area:

1. New event detection (NED)
   This challenge occurred when we don't know what to expect. NED real-time system should correctly identify new event that has not been discussed before.
2. Event Tracking
   Refers to the study of how events spread and evolve.
3. Event summarization

Creating summary about events based on bursty features.

4. Event Associations
   Events could impact on each other. Event associations based on analysis of the events relationships. It is also denoted as ”event graphs” [7].

Another survey [8] includes research on event detection about disasters, news, out-breaks and traffic. The authors' work was focused on analysing only the Twitter OSN. Authors mentioned actual work on pandemic outbreak detection made in [9] work. In Twitter people share information about their life, something personal. In case of pandemic users could post information about their health, symptoms or share any medical information. This data could be very helpful for medical authorities to estimate epidemic scale.

The natural disasters is a common interest of various researchers. People post essential information during and after event according to their feelings and real-time situation. Research [10] concentrated on the evolution of rare event, like storm, in the real world by analysing activities in virtual world. Their case study is based on Hurricane Sandy in 2012. Authors denotes the idea for social media activities temporal pattern construction.

Event detection technique depends on event characteristics and its category. Authors of [11] made a survey on event detection techniques for natural disaster events, trending topics and public opinion events in newswire, web forums, emails, blogs and microblogs. They defined that domain dependence is a huge challenge for researchers because techniques are extremely situational dependent. Time constraint is another characteristic which is also varying according to event category. Authors discussed techniques grouped by information flow between users: thematic, temporal, spatial and network structure. The survey includes researches on OSNs like Twitter, Facebook, Instagram, Youtube and Pinterest.

A huge survey [12] proposes definitions and categorizations in event detection process. The authors classified 34 works by event types, pivot techniques, detection method, detection task and application. The majority of researches were for detecting general interest events.

Researchers highlight relevance of event detection:

- “Event detection contains substantial information which describes different scenarios during events or crisis. This information further helps to enable contextual decision making, regarding the event location, content and the temporal specifications.”[11]
- “Could be used as an emergency notification tools about a disaster that coming on, as reference to manage traffic, as prediction tool for flu trends.”[8]
- ”Better event description through sub-event detection”[6]

3. Cloud-Native Application

Since social media crawler works with large amount of continuously increasing amount of data, it has several requirements for proper work.

1. Different Online Social Networks. Since social media crawler should work with different OSNs then external APIs will be different. However internal data processing will be identical for all.
2. New OSNs addition. This process should be gentle and simple. New resources should be given for additional OSN.
3. Data pre-processing and analysing. Increasing data cause increasing of preprocessing and analysing time.

In this section we are describing technology that fits our system requirements.

Elasticity is the degree to which the system is able to adapt to changes in workload by providing and removing resources in an autonomous approach, such that at any given time, the available resources are as close as possible to current demand [13].

Scalability. It can be differentiated into “structural scalability and load scalability. Structural scalability is the ability of a system to expand in a chosen dimension without significant changes in its architecture. Load scalability is the ability of the system to work correctly when the offered traffic increases [14].

Self-service deployment - is understood as part of the application deployment topology to implement a specific technical unit [15]. More often, a unit of deployment is understood as a “standard con-

tainer". The goal of a standard container is to encapsulate a software component and all of its dependencies in a self-describing and portable format so that any compatible runtime can run it without additional dependencies, regardless of the underlying machine and the contents of the container. This is a definition from the Open Container Initiative (OCI), which is explained in 5 principles of standard containers [16].

Today there is a paradigm of software development, which lays in business processes and architecture solutions to the above issues. The paradigm is called Cloud-Native Application Development (CNA Development) [17]. This term refers to a set of technologies and design patterns that have become the standard for building large-scale cloud applications. Software development in this paradigm provides the properties of successful cloud applications, including dynamic scalability, ultimate resiliency, non-disruptive upgrades, and security. To enable the creation of applications that meet these requirements, we describe a microservices architecture that is central to cloud design.

The author of work [18] analysed more than 50 works related to the development of cloud applications, collected and summarized approaches, methods and terms. As a result, they define the term Cloud-Native Application as follows: "A cloud application (CNA) is a distributed, flexible and scale-out system of (micro) services that isolates state in a minimum of stateful components. The application and each individual deployment module of this application are designed according to cloud-centric design patterns and run on a flexible self-service platform. "

In [19], it was proposed to call such applications IDEAL, so that the application was [Isolated state] isolated, [Distributed] had a distributed architecture, was [Elastic] flexible in the sense of horizontal scaling, was controlled using [Automated] automated systems, and its components must be loosely coupled. Creation of cloud applications in this paradigm leads to the following results [20]:

- faster provision of software solutions to the customer
- fault tolerance
- automation of recovery
- easy and fast horizontal scaling of applications
- the ability to process a huge amount of data

4. Results

In this section we will describe our architecture for social media data collection and processing, Telegram as OSN case study and database structure.

4.1. System architecture

During this stage of the research we have developed a cloud architecture based on microservices. Microservices are the decomposition of monolithic business systems into independently deployable services that perform a single task. The main way to communicate between services in a cloud application architecture is through published and versioned APIs (API-based collaboration). In our architecture microservices communicate via message queue.

The individual architecture deployment units are designed and interconnected according to a set of cloud-oriented patterns such as a twelve-factor app [21], a Circut Breaker [22].

We use the flexible OpenStack platform, which is used to deploy and operate these microservices through autonomous deployment units (containers). This platform provides additional operational capabilities on top of IaaS infrastructures, such as auto-scaling application instances and scaling on demand, application health management, dynamic routing, load balancing, and log and metrics aggregation.

In Figure 1 and Figure 2 we depicted the parsing process flow through our cloud architecture and general view of architecture. On that figure microservices have symbol of '*' near their names.

New message created in OSN detected and parsed by crawler. Firstly, crawler will save message in database. Secondly, it will put textual content of the message in queue. Queue service has an exchange area and defined queues. Tokenization consumer receive its data through queue. Consumer is located in data processing microservice. This microservice will be exposed later with other processing tools. After data pre-processing and processing, all extracted information go to database.

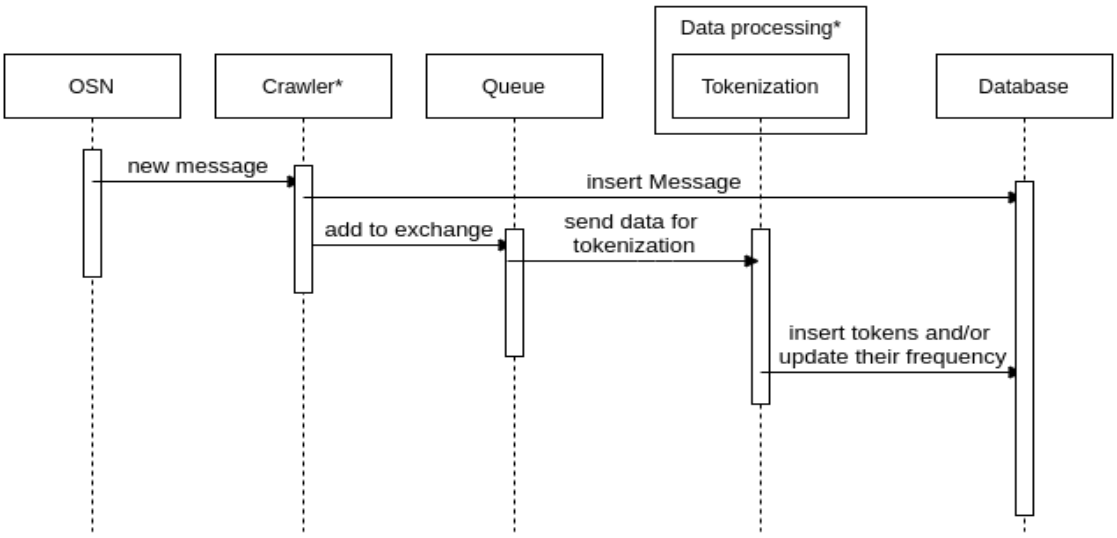

Figure 1. Parsing process flow

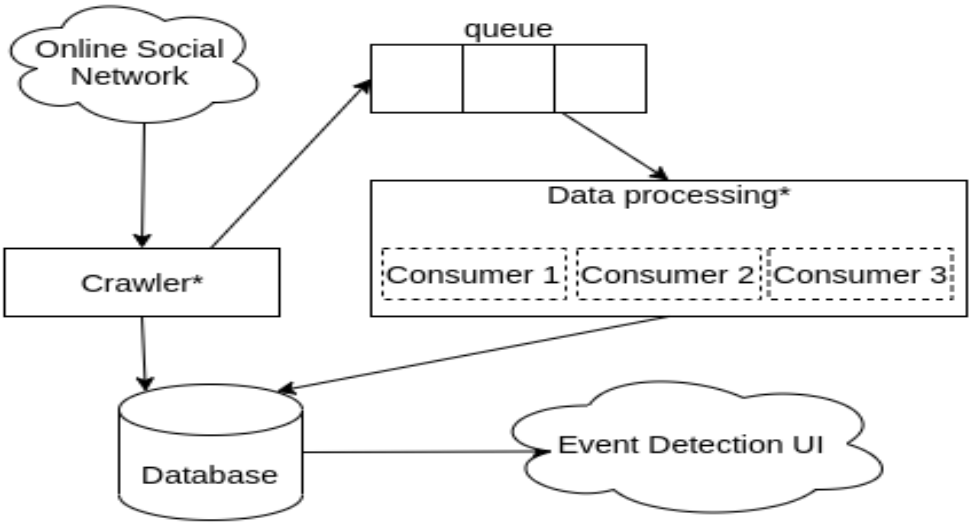

Figure 2. Architecture

4.1. Telegram

Nowadays social networks developers usually provide public tools or libraries to interact with their system and its data. It was mentioned above that Telegram becomes more and more popular. We decided to construct our social media crawler firstly based on Telegram OSN.

Telegram is an OSN which provides several communication ways. Users can communicate with each other in private chats, tet-a-tet, or create chat for more than two users. In telegram people can create channels where they post anything and other users may subscribe on them. Channels are similar to newswires.

According to official Telegram Database Library (TDLib) provided by Telegram, we have constructed our Client API application [4]. This application works under Telegram API Terms of Service and needs real registered user in original Telegram. Client API needs application authorization. We have created application in Telegram with real registered phone number. Application Id used for authorization in Client API. In case of breach of terms, the access to Telegram API will be discontinued via application id. Client API handles all the activities of authorized user. Client was added to public groups and channels of different interest like IT, politics, news, cinema, money, psychology. New

messages and posts go through our application and go to database. Client API written on Java. TDLib is written on C, but it already has native Java (using JNI). For database we used PostgreSQL.

4.2. Database architecture

TDLib allows to collect a lot of information about users, chats and messages. The Figure 3 shows relational model diagram of our database.

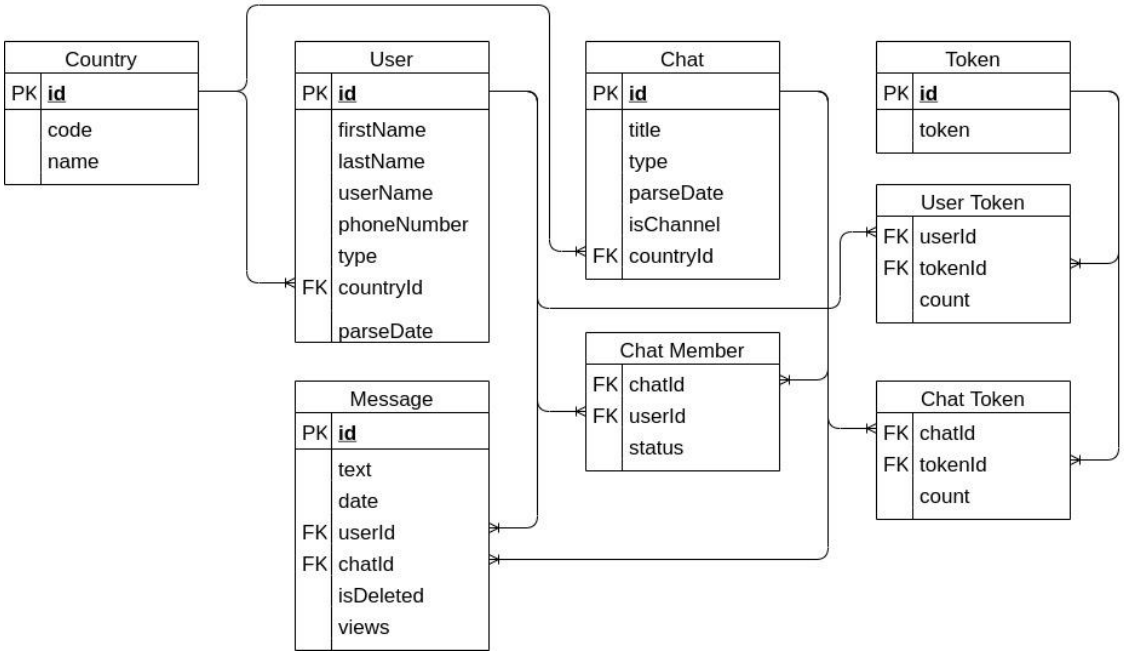

Figure 3. Database structure

The first implemented preprocessing task was message text tokenization. The token in our case is unigram. Before tokenization, text is cleared from Russian stop-words. Each token has its count within users' messages and chats' messages. For example, in Kazakhstan during COVID-19 pandemic and state of emergency, government pays 42500 tenge of compensation for people who lost their job or income. This '42500' token has total count greater than 9000 among chats, but it was mainly forced by conversations in one news channel. The biggest count among users is 194.

From 19.02.2020 we are collecting messages from public groups and channels from Kazakhstan. At the beginning of April we included chats from Russia, Belarussia, Ukraine and Uzbekistan. On 25.10.2020 database consists of 2141 chats, 242 000 users and 840 648 messages

5. Conclusion

We have developed highly loaded, fault-tolerant, scalable system for social media data collection using Cloud Native Application Development paradigm. Crawler collects messages from 2141 Telegram chats. Data processing microservice designed as scalable unit which could be easily exposed with additional tools. In future works we will detect events and go further to our main research goal.

References

1. Klepek, M. and Starzyczná, H. (2018). Marketing communication model for social networks. Journal of Business Economics and Management, 19:500–520.
2. Sharma, S. and Verma, H. (2018). Social Media Marketing: Evolution and Change, pages 19–36.
3. Telegram channels kazakhstan. https://kaz.tgstat.com/ Visited: 2020-04-18.
4. Telegram database library. Version 1.6.0. Jan 31, 2020. https://core.telegram.org/tdlib. Visited: 2020-02-10.

5. Dou, W., Wang, X., Ribarsky, W., and Zhou, M. (2012). Event detection in social media data. Proceedings of the IEEE VisWeek Workshop on Interactive Visual Text Analytics-Task Driven Analytics of Social Media Content, pages 971–980.
6. Saravanou, A., Katakis, I., Valkanas, G., and Gunopulos, D. (2018). Detection and delineation of events and sub-events in social networks. pages 1348–1351.
7. Yang, C., Shi, X., and Wei, C.-P. (2009). Discovering event evolution graphs from news corpora. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 39:850 – 863.
8. Nurwidyantoro, A. (2013). Event detection in social media: A survey. pages 1–5.
9. Ritterman, J., Osborne, M., and Klein, E. (2009). Using prediction markets and twitter to predict a swine flu pandemic. 1st International Workshop on Mining Social Media.
10. Lu, S., Zhou, M., Qi, L., and Liu, H. (2019). Clustering-algorithm-based rare-event evolution analysis via social media data. IEEE Transactions on Computational Social Systems, PP:1–10.
11. Goswami, A. and Kumar, A. (2016). A survey of event detection techniques in online social networks. Social Network Analysis and Mining, 6.
12. Cordeiro, M. and Gama, J. (2016). Online Social Networks Event Detection: A Survey, volume 9580, pages 1–41.
13. Herbst, N.R., Kounev, S., Reussner, R., 2013. Elasticity in cloud computing: what it is, and what it is not. In: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13). USENIX, San Jose, CA, pp. 23–27.
14. Bondi, A.B., 2000. Characteristics of scalability and their impact on performance. In: Proceedings of the 2nd International Workshop on Software and Performance. ACM, New York, NY, USA, pp. 195–203. doi:10.1145/350391.350432.
15. Inzinger, C., Nastic, S., Sehic, S., Vögler, M., Li, F., Dustdar, S., 2014. Madcat: A methodology for architecture and deployment of cloud application topologies. In: Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on, pp. 13–22. doi:10.1109/SOSE.2014.9.
16. Open Container Initiative, 2020. Open container runtime specification. https://opencontainers.org/ Visited: 2016-05-27
17. Gannon, Dennis, Roger S. Barga and Neel Sundaresan. “Cloud-Native Applications.” IEEE Cloud Computing 4 (2017): 16-21.
18. Kratzke, Nane, and Peter-Christian Quint. “Understanding Cloud-Native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study.” Journal of Systems and Software 126 (2017): 1–16. https://doi.org/10.1016/j.jss.2017.01.001.
19. Fehling, C., Leymann, F., Retter, R., Schupeck, W. and Arbitter, P., 2014. Cloud computing patterns: fundamentals to design, build, and manage cloud applications. Springer Science & Business Media.
20. Stine, M., 2015. Migrating to Cloud-Native Application Architectures. O’Reilly.
21. Adam Wiggins, 2014. The twelve-factor App. http://12factor.net Visited: 2020-02-14.
22. Martin Fowler, 2014. Microservices - a definition of this new architectural term. http://martinfowler.com/articles/microservices.html Visited: 2020-05-20.

СОДЕРЖАНИЕ