About the User Classification Problem Based on Analyzing the Odnoklassniki Friendship Graph

Alexey Zinoviev, PhD student, OmSU

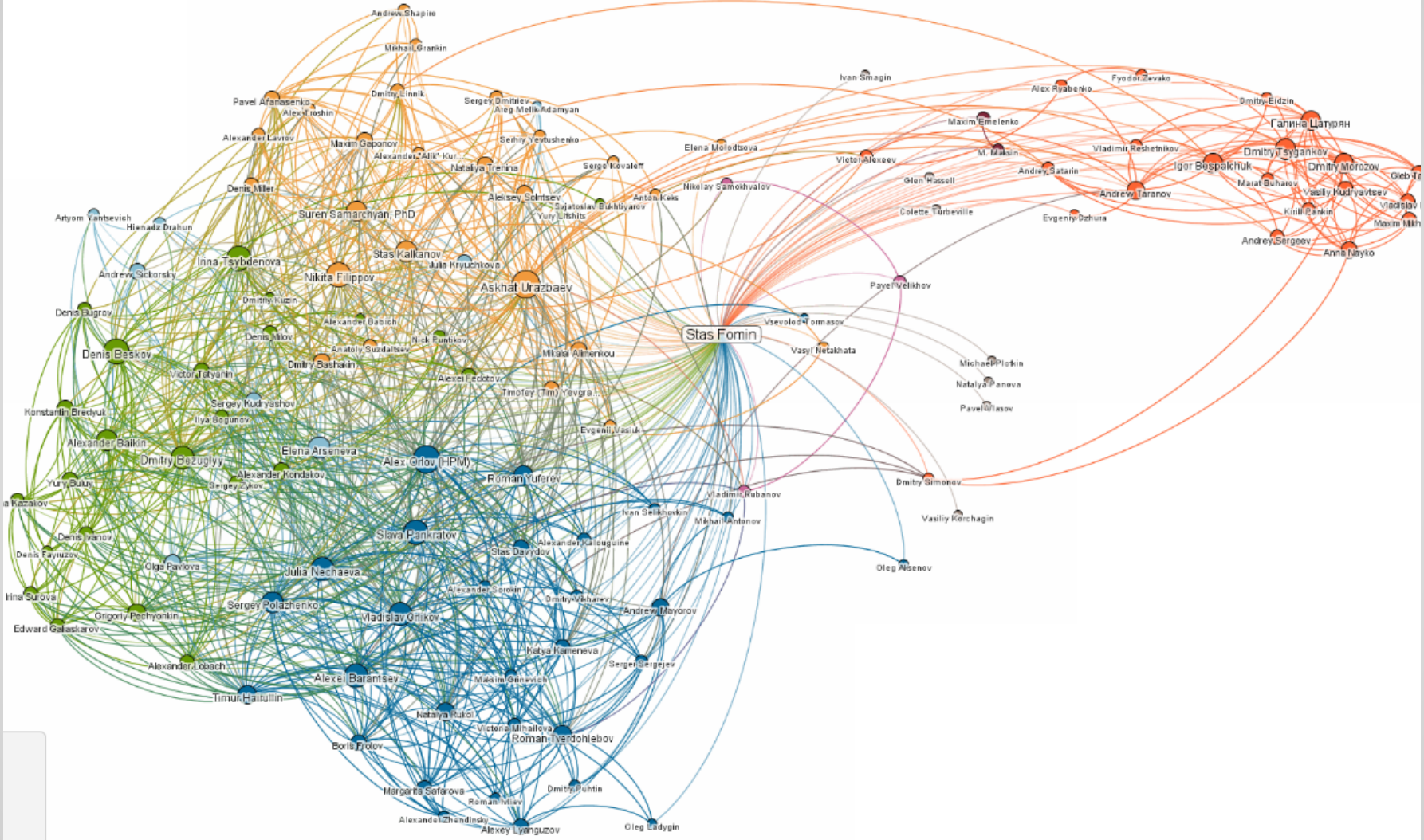
Social Network

In common:

- 200 000 000 users
- 8 500 000 communities

Per day:

- 40 000 000 users
- 250 000 000 messages
- 8 000 000 posts
- 12 000 000 photos
- 7 000 000 new links (friendships)


Andrew Shapiro
Mikhail Gronkin
Pavel Afanasenko
Alexei Ioshin
Dmitry Linnik
Sergey Dmitriev
Oleg Melik-Adamyan
Alexander Lavrov
Maxim Gaponov
Alexander "Alik" Kur
Sergey Yevtushenko
Natalya Tremina
Sergei Kovalev
Denis Miller
Alexey Schitsev
Sviatoslav Bukhtiyarov
Yury Litshits
Anton Keks
Nikolay Samokhvalov
Artyom Yantsevich
Hienadz Drahun
Suren Samarchyan, PhD
Stas Kalkanov
Julia Kryuchkova
Irina Tsybdenova
Andrew Sickorsky
Nikita Filippov
Askhat Urazbaev
Dmitry Kuzin
Denis Bugrov
Alexander Babich
Denis Milov
Nick Puntikov
Anatoly Suzdaltsev
Denis Beskov
Victor Tolyanin
Dmitry Bashkirin
Mikalai Alimenkou
Alexei Fedotov
Timofey (Tim) Yevgrafov
Sergey Kudryashov
Konstantin Bredyuk
Ilya Bogunov
Alexander Balkin
Dmitry Bezuglyy
Elena Arseneva
Alex Orlov (HPM)
Roman Yuferev
Alexander Kondakov
Sergey Zykov
Yury Buluy
Evgeni Vasiliev
Denis Ivanov
Denis Fayruzov
Olga Pavlova
Julia Nechaeva
Irina Surova
Sergey Polazhenko
Grigoriy Pechyonkin
Edward Galaskarov
Slava Pankratov
Alexander Sorokin
Dmitry Vikharev
Vladislav Grikov
Alexander Lobach
Alexei Baramtsev
Timur Hairullin
Katya Kameneva
Sergei Sergejev
Maksim Grinevich
Natalya Rukol
Victoria Mihailova
Roman Iverdohlebov
Boris Frolov
Margarita Seferova
Roman Iviev
Dmitry Puhtin
Alexander Zhendinsky
Alexey Lyanguzov
Oleg Ladygin
Stas Davydov
Alexander Kalouguine
Ivan Selikhovkin
Mikhail Antonov
Vladimir Rubanov
Dmitry Simonov
Vasiliy Kerchagin
Oleg Alsenov
Pavel Velikhov
Vsevolod Tomasov
Vasyl Netakhata
Stas Fomin
Michael Plotkin
Natalya Panova
Pavel Klasov
Ivan Smagin
Alex Ryabenko
Fyodor Zevako
Maxim Emelenko
Elena Molodtsova
Victor Alexeev
M. Maksin
Glen Hassell
Colette Turbeville
Evgeny Dzhuta
Andrey Satarin
Andrew Taranov
Vladimir Reshetnikov
Igor Bespalchuk
Dmitry Tsygankov
Dmitry Eidzin
Галина Шапурян
Dmitry Morozov
Marat Buharov
Vasily Kudryavtsev
Kirill Rankin
Andrey Sergeev
Anna Nayko
Vladislav
Maxim Mikh
Gleb

The malicious activity

- Offense against the laws of ethics, morality, and articles RF Criminal Code
- Creation of hidden subnetwork with spam accounts
- Hacking profiles of actual users
- Spam attack from hacked profiles
- Attraction of user’s attention by user’s page visiting

The benefits of social network

- Prevent the spread of profiles breaking "epidemic" oand leakage of personal data
- Prevent spam before it arrives
- Reduce the number of complaints
- Reduce the burden on moderators
- Reduce the moderator staff

Dataset

- Graph (~ 9 * 10^6, 39 Gb)
- Demography
- User likes
- History of logging (~ 3,2 * 10^8, 12 Gb)
- Community posts
- Complaints about spam

Tools

- R 3.0.3 (for prototyping only)
- python 23 + scypi + numpy + pandas (data mining)
- Hadoop 2.6 (cluster infrastructure)
- Pig 14 (for user’s features calculating)
- Giraph 1.1 (for graph-related features calculating)

The Problem

It should offer mathematical model makes prediction with high reliability to determine that the user is an attacker. It should be based on the number of friends, history of logging and analysis of other activities (type I error is not more than 1% and a type II error < 10%) .

The model

Set of objects - social network users

Each object should be classified as User or Spamer.

Training set is produced on complaints of actual users.

Features

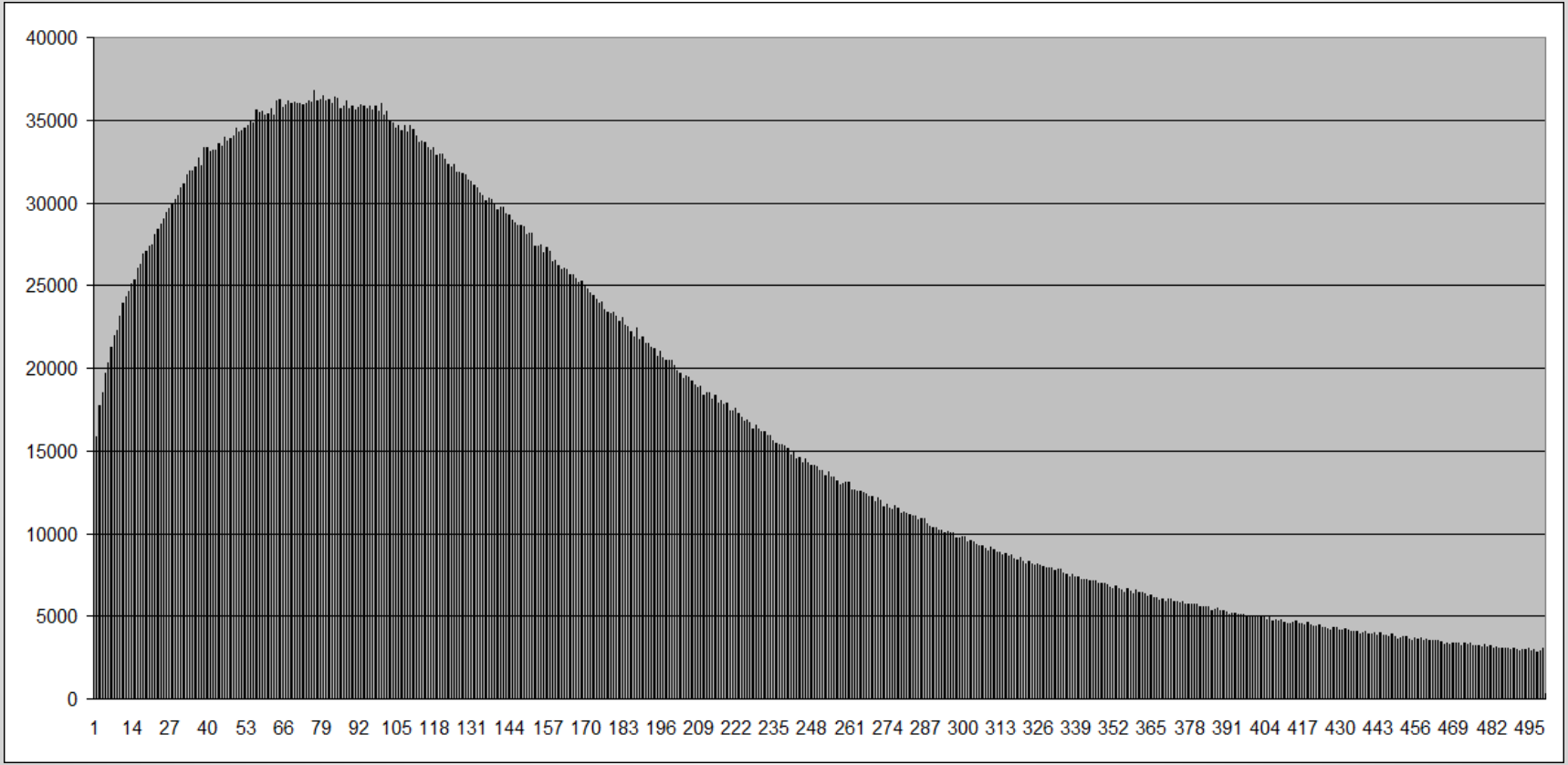
- Local feature: vertex degree
- Global feature: PageRank for each vertex
- Global-local feature: local clustering coefficient value (LCC)
- Number of successful logins
- Demography
- Geography

Training set

Features were calculated for 10000 users:

- age, is_male, is_female
- degree, lcc, page_rank, geo_lcc
- good_auth_per_week, bad_auth_per_week
- dist_from_Moscow, dist_from_borders

Vertex degree distribution



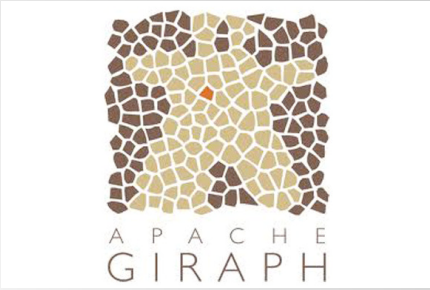
Computational experiment

4 servers with 8 cores and 30 Gb RAM, in Google Compute Engine.

Hadoop Cluster + Pig for feature calculation.

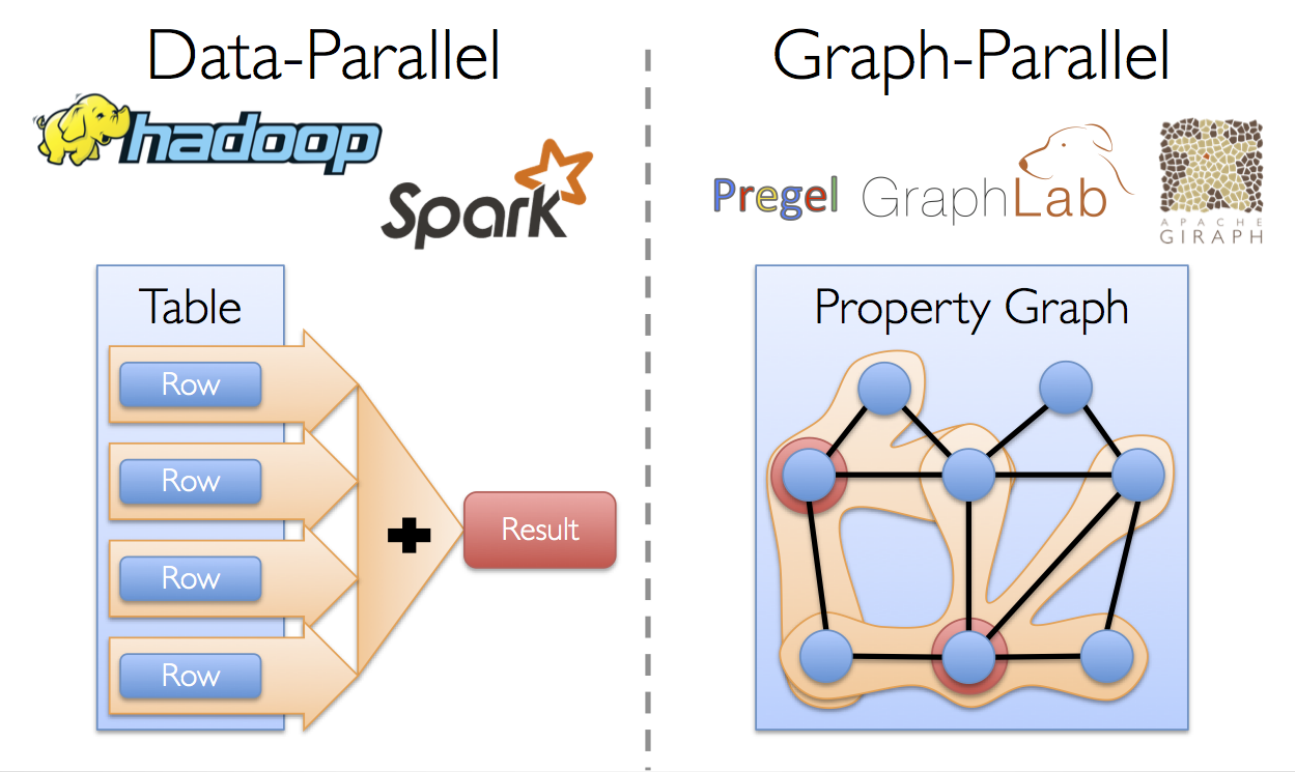
Giraph, above Hadoop cluster for calculating of PageRank and lcc.

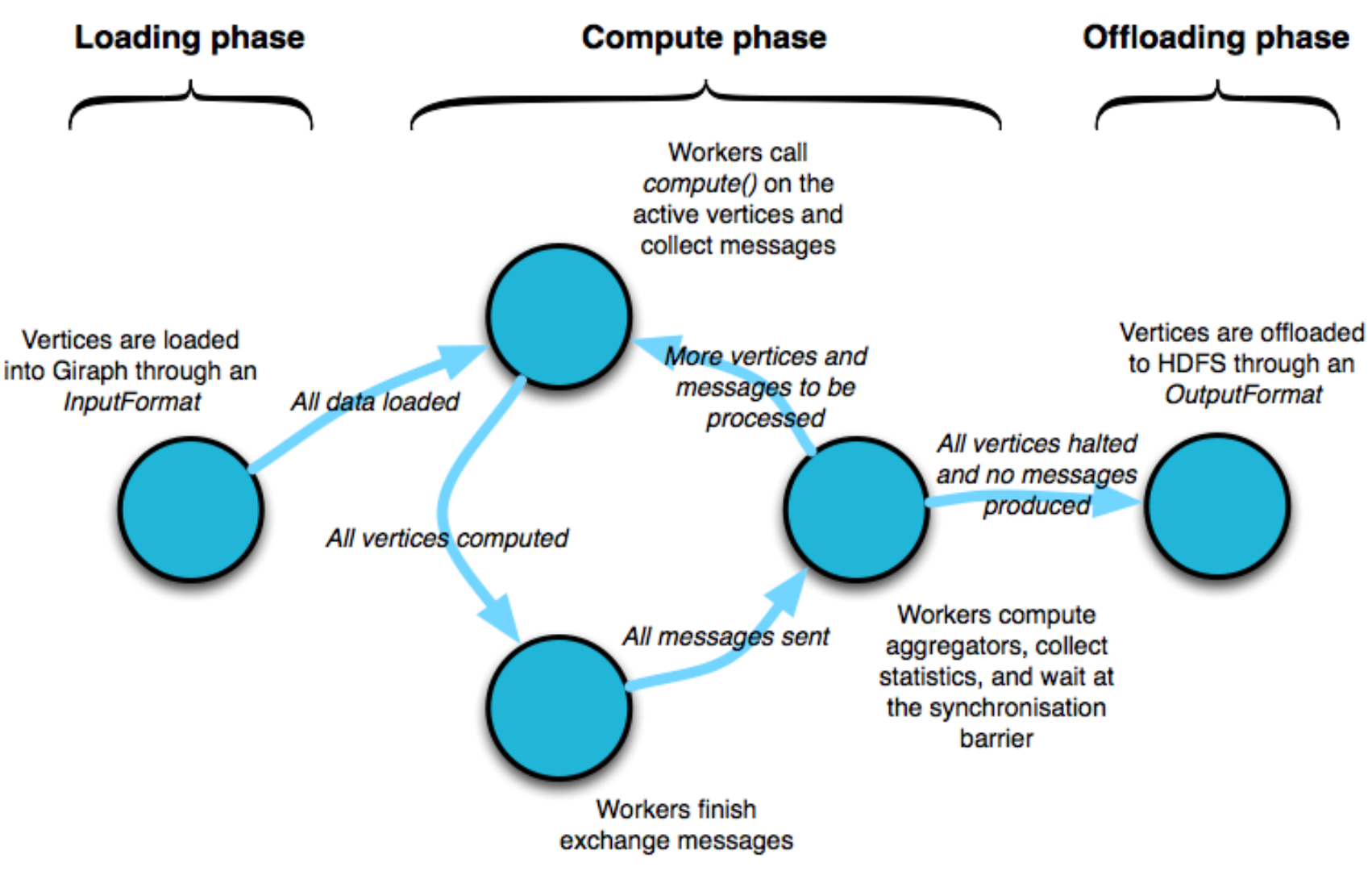
Why Giraph?



- Open-source Pregel implementation
- Works on existing Hadoop infrastrucure
- Calculations in memory
- Simple organized iterative calculations (it’s important for PageRank)

Думай вершинами, а не строками...




Loading phase
Compute phase
Offloading phase
Workers call compute() on the active vertices and collect messages
Vertices are loaded into Giraph through an InputFormat
All data loaded
More vertices and messages to be processed
Vertices are offloaded to HDFS through an OutputFormat
All vertices halted and no messages produced
All vertices computed
All messages sent
Workers compute aggregators, collect statistics, and wait at the synchronisation barrier
Workers finish exchange messages

Time of experiment

Iterative execution of PageRank, written in Pig was finished in 25 iterations, 123 minutes (~ 5 minutes per iteration)

Giraph implementation of PageRank cost 45 iterations and 25 minutes (~ 35 seconds per iteration) with running condition 1 worker per 1 core

Model

For model creation it used kNN, polynomial regression and decision trees(Random Forest, C4.5).

The best results had kNN (n = 7) and C4.5 with type I error 5% and 3%, type II error 12% and 19%, respectively.

Feature’s importance

geo_lcc and **degree** are most important features, after followed in order of importance l**cc, dist_from_Moscow, good_auth_per_week** and **page_rank**.

But social-demography data provided by each user in his personal profile had a worst importance in decision trees and low importance for kNN.

In conclusion

- Calculation of graph features for big dataset is very difficult for MapReduce approach and needs in Pregel approach.
- Features derived from the analysis of the relationship structure are important in solving the problem of spam accounts searching.
- Hadoop + Pig + Giraph in Google Compute Engine - easy scalable infrastructure for implementing SNA models and algorithms.

Haec habui, quae dixi