Geo2Tag performance evaluation

Mark Zaslavskiy, Kirill Krinkin
FRUCT LETI Lab,
Open Source & Linux Lab

FRUCT-12, Oulu, November, 2012
Motivation

- Geo2Tag Location-Based Services platform was started as project for students who want to get an Open Source development experience.
- With increasing number of functions which are supported by platform performance requirements also increased.
- Platform performance evaluation was never performed.
Goals

- Investigate the most frequent REST requests to platform performed by users;
- Measure performance (processing time) of the most frequent requests;
- Determine bottlenecks in request processing;
- Maximize performance of request processing and DB synchronization;
Platform architecture

Internet

HTTP/JSON

Platform Interface

Lighttpd

FastCGI

Control program

PostgreSQL

qSQL

DB

Request processing

Interaction with the database
Client-server interaction modeling

• The most frequent REST requests were found using modeling of LocationClient app behavior
• Modeling was performed using Markov chains theory
• Request executions were used as non-absorbing states of Markov chain
• Client application shutting down was used as absorbing state of Markov chain
Client-server interaction modeling

- During numerical experiments average number of each request executions before absorption were calculated.
- Modeling was performed for different initial states and tagging frequency.
- As a result of modeling \texttt{WriteTag} and \texttt{LoadTags} requests had the biggest average number of executions for all conditions.
Control program profiling

- For determining bottlenecks during request processing profiling performed
- For each write request whole processing time and DB interaction time were measured
- In average DB interaction takes ~95% of request processing time
- DB interaction is bottleneck!
Control program profiling

![Chart showing time in seconds for different operations]

- AddChannel
- AddUser
- DeleteUser
- RegisterUser
- SubscribeChannel
- UnsubscribeChannel
- WriteTag

- Total request processing time
- DB interaction
Platform optimizations

- DB synchronization optimizations:
  - Thread synchronization refactoring (less locks – more performance)
  - Algorithm for making decisions about need of DB synchronization
- DB structure optimization
  - Number of tables for tags storage were decreased to one – **WriteTag** processing speed increased
DB synchronization

• Algorithm which determines does system need DB synchronization or not
  – Periodical check with user-defined period (UPDATE_INTERVAL)
  – Comparison between actual number of transactions to DB and transactions number counted by platform
  – If difference is more than TRANSACTION_DIFF value than synchronization is performed, else nothing happens
  – Variation of UPDATE_INTERVAL and TRANSACTION_DIFF allows to achieve different ratios “performance/data consistency”
Request performance measurement

- Performance were measured for **WriteTag** and **LoadTag** requests in system before and after optimization
- Processing time and errors data were collected
- For **LoadTag** set of experiments were performed with different number of tags in DB – from 0 to 54000 with step of 1000 tags. For each tag value 10000 **LoadTags** requests where performed
- For **WriteTag** performance measurement was performed as sequentially increase of tags number in system from 0 to 12000 by sending **WriteTag** request
Results comparison

• Maximum, minimum, average, variance of request processing time and number of errors where compared for both systems
• **LoadTags**
  – Maximum processing time decreased
• **WriteTag**
  – Average time decreased by 47.5%
  – Maximum time decreased tenfold
  – Variance of time decreased hundredfold
Conclusions

• During performance evaluation where achieved next results:
  – Math model of client application was created
  – Bottleneck of request processing was found
  – Performance of request processing was optimized
• Future plans:
  – Support NoSQL or GIS-oriented DB
  – Support lock-free algorithms and data structures
Questions & Answers

Mark Zaslavskiy, Kirill Krinkin
{mark.zaslavskiy, kirill.krinkin}@gmail.com
Open Source & Linux Lab,
http://osll.furct.org, osll@fruct.org