Department of Informatics

University of Zürich
Binzmühlestrasse 14
CH-8050 Zürich
Switzerland

Prof. Dr. Burkhard Stiller
Andri Lareida
Dr. Thomas Bocek
Communication Systems Group
CSG@IFI
Phone: +41 44 635 4396 (Supervisor)
Fax: +41 44 635 6809
E-mail: stiller@ifi.uzh.ch
tsiaras@ifi.uzh.ch
bocek@ifi.uzh.ch
URL: http://www.csg.uzh.ch/

Master Thesis for Oliver Zihler

Task Description:	Dr. Thomas Bocek
Title:	**MapReduce for Big Data with a DHT**
Start Date:	October 26, 2015
End Date:	April 26, 2016
Supervisor:	Dr. Thomas Bocek, Patrick Poullie
Location:	IFI, Home
Support:	TomP2P, Java, CSG testbed, evaluation, P2P, MapReduce

1. Introduction and Motivation

P2P systems are still popular and account for a large portion of Internet traffic [5]. Most of this P2P traffic is related to file sharing (BitTorrent [6] (BT)), but also new types of P2P applications are emerging. Another trend is that the number of devices connected to the Internet is increasing [8], especially mobile devices [7]. Therefore, synchronization between these devices becomes more important since users tend to have more than one device on which data is accessed, modified, or created. The basis of every P2P network is that it consists of equitable peers that are interconnected. In contrast to client-server architecture, every peer in a P2P system can be server *or* client.There are three types of P2P networks: structured, semi-structured, and unstructured. In the beginning of the P2P network development and research, Napster and Gnutella determined the start. The former showed an approach with central elements within a single point of failure. The latter was unstructured and did not scale very well. An introduction of super peers made the unstructured network semi-structured [1]. A super peer is able to do more than other peers in terms of resources or network properties. The most recent research focuses on structured networks. With structured networks [2], [3], such as distributed hash tables (DHTs), resources can be found in *O(log n)*, where *n* is the number of peers in the system.

For several projects, TomP2P [4] for the underlying DHT has been used. TomP2P uses an XOR metric, similar to Kademlia [3] and implements a distributed hash table (DHT), an extended DHT that supports besides *put(key, value)* and *get(key)* operations, also *add(key, value)*, and *digest(key)* operations and other operations. On top of TomP2P, MapReduce has been alrayd prototyped in version 4, however, with minimal functionality. Thus, the goal of this thesis is to add MapReduce to the lastest version 5 that can be used for Big Data processing.

2. Description of Work

A DHT is fault-tolerant distributed system that stores key-value pairs. The goal of this thesis is to use a DHT as a building block to allow MapReduce operation. This thesis will build a prototye of MapReduce using a DHT as a basis.

To build such a prototype, the feasibility of MapReduce on top of a DHT has to be evaluated. Therefore, detailed knowledge about the DHT implementation is required to analyze the effects of typical MapReduce operations in the underlying DHT.

Based on the findings of the feasibility analysis a prototype design and implementation will be produced. The prototype shall allow the basic MapReduce operations including redundancy and support for JavaScript-based tasks using the Nashorn scripting environment. Further aspects that need to be considered are resource allocation to not overload a peer and resource isolation (e.g. cgroup as used with Hadoop).

The prototype will then be used in an evaluation effort to prove the findings form the initial analysis. The evaluation should compare the implementation to existing MapReduce frameworks such as Hadoop [9] and compare its finding with the results in [10].

3. Master Thesis Goals

Driven by the description of work outlined, the following key goals are suggested for this Master Thesis. The final goals will be discussed during the thesis because of the explorative nature of this thesis:

- Design, Implement, and evaluate MapReduce on top of a DHT
- Design the API in such way that the developer has high flexibility with minimal input parameters. The API needs to be developer friendly and easy to understand.
- Set automatically as much parameters as possible to an optimal value. These values should be also automatically adjusted during runtime to offer the best performance (including resource allocation).
- Use testcase (JUnit) and implement a testing framework.
- Documentation and developer guide for the library. Source code needs to be well documented and readable
- Evaluate the prototype in different scenarios (differend number of machines, heterogenous hardwaro, different languages - Java/Javascript) using the “word count” example as a basis
- Write a well-structured report that shows your findings

4. Activities

Based on the description of work, the following tasks targeting the required milestones need to be accomplished:

- Milestone 1: Planning
  Discuss your initial planning with the supervisors.
- Milestone 2: Analysis
  Analyze what tasks are suitable for MapReduce besides “word count” and find suitable big data for processing
- Milestone 3: Design
  Prototype design is finished. Decision on API design is made.
- Milestone 3: Running prototype
  Prototype implementation has the main features, minor bugs are acceptable.
- Milestone 4: Prototype completed
  The prototype works. Plan for the evaluation in a testbed.

- Milestone 5: Evaluation completed
  Run experiments in the testbed with multiple machines and gather data to support your findings.
- Milestone 6: Hand in and final presentation.
  Evaluation done and thesis written.

5. General Notes

- Students have to provide a written schedule for their full thesis steps within the first two weeks of their work. Clarify details with your supervisors and finalize the schedule of tasks, basically in a weekly fashion. Include the intermediate/public presentation(s), too.
- Prepare an intermediate and internal presentation (20 min max plus Q&A) after half time of your thesis (date to be set) and discuss this with your supervisors. At the thesis end (date to be set) a final public and self-contaning presentation (20 min max plus Q&A) has to be given.
- The final written report will document all work undertaken and remember that this report must be self-contained. Major technical basics are to be included and detailed knowledge obtained during the work must be documented. Assumptions, design decisions, configuration choices, and results are part of the report as well as implementation details and usage information. Correct bibliographic references and a list of papers, recommendations, and descriptions used must be added, including those ones given below.
- All software develoment activities must adhere to the standards of sofwtare engeneering. This includes but is not limited to the use of UML, Unit tests, dependency management, and version control. If uncertainties of which tools can be used exist, the supervisors will clarify.
- Establish periodic meetings with he supervisors to report on progress and to discuss problems.
- Students involved in this thesis are required to read and answer e-mails related to this project at least three times a week. A more frequent, if needed, interaction will determine a better basis for supervision and progress.
- The information sheet on important hints for thesis work is required to be known.

6. Formal Results

Besides the intermediate and final oral presentation of your work the following written documents are part of your work and need to be handed in to the supervisors in time:

- A printed, double-sided report in a soft cover binding and in 3 copies (in English or German): It covers the motivation and introduction, the problem to be solved, the discussion of the design choices, a set of arguments on the final design choice, a list of solved and open issues. A critical consideration of the task, the work, and the result will conclude the report. Additionally, a table of content and figures (including tables), a valid list of bibliographic references, and optional appendices as required is part of the report, too. The official acknowledgement section is mandatory, a personal one optional, however recommended, as usually a number of people took part in the process of finalizing the thesis. The text processing shall be done in FrameMaker (preferred) or LaTeX and in rare cases in Word.
- A documentation of the implemented/configured system, covering the system’s view point, an implementation description, a use and installation manual, a documentation of all program and data structures developed or utilized is essential. All of this may be part of the report above, however, in case it will not be included, it must exist in a separate document.
- A dedicated CD for each copy of the report has to be produced: (1) a collection of all program sources, software components, protocol implementations utilized, hardware/product documentation in PDF, related work papers in PDF or full HTML pages, and a directory description; (2) the written thesis (report) in source files, figures in source file and jpg or eps, and a full printable PS as well as PDF file, the set of slides for the final presentation in PowerPoint; and (3) all further mate-

rial used, if available in electronic form, such as all existing and documented test scenarios, testing plans, and test results.

- A German abstract (Kurzfassung) in a maximum of 2 pages (in case of a report written in German, this abstract needs to be in English), which will enable a quick and clear survey of this work, tasks and results. This summary will be part of the bound report, and is included after the front page and before any other text will follow. It includes four short sections on: 1. Einleitung/Introduction, 2. Ziele/Aims and Goals, 3. Resultate/Results, and 4. Weitere Arbieten/Further Work.
- The complete set of copies of the report, the CD, and the talk must be completed and handed in to the supervisors in time before the thesis will change into the "submitted" status. Note that failures of delivering those information and data after the formal hand-in time may result in a “non-passing” state of this work.

7. References

[1] Yatin Chawathe and Sylvia Ratnasamy and Lee Breslau and Nick Lanham and Scott Shenker, *Making Gnutella-like P2P Systems Scalable,* In Proceedings of the ACM SIGCOMM '03 Conference, Karlsruhe, Germany, August 2003

[2] I. Stoica, R. Morris, D. Karger, F. Kaashoek and H. Balakrishnan: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. Proceedings of the 2001 ACM SIGCOMM Conference, San Diego, California, USA, pp 149-160, August 2001

[3] P. Maymounkov and D. Mazieres: Kademlia: A peer-to-peer information system based on the XOR metric. in Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, MA, USA, March 2002

[4] TomP2P: http://tomp2p.net, last visited: 12.10.2015

[5] CISCO. (2013, Jan.) Cisco Visual Networking Index: Forecast and Methodology, 2011-2016. [Online]. Available: http://www.cisco. com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ ns827/white paper c11-481360 ns827 Networking Solutions White Paper.html

[6] B. Cohen, “Incentives Build Robustness in BitTorrent,” in 1st Workshop on Economics of Peer-to-Peer Systems (P2PECON), Berkeley, CA, USA, June 2003.

[7] Lareida et al. (2013). Box2Box - A P2P-based File-Sharing and Synchronization Application

[8] M. Meeker and L. Wu. (2013, May) Internet trends. [Online]. Available: http://allthingsd.com/tag/ mary-meeker/

[9] Hadoop, https://hadoop.apache.org/, last visited: 12.10.2015

[10] L. A. Steffenel and M. Kirsch-Pinheiro, “CloudFIT, a PaaS platform for IoT applications over Pervasive Networks” in 3rd International Workshop on CLoud for Iot (CLIoT), Taorina, Italy, September 2015

Zürich, October 26, 2015

Prof. Dr. Burkhard Stiller