2015 7th IEEE International Workshop on Performance Evaluation of Communications in Distributed Systems and Web based Service Architectures

Extensive Analysis of the Kad-based Distributed Computing System DuDE

Peter Danielis, Jan Skodzik, Vlado Altmann, Benjamin Kappel, Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany, Tel./Fax: +49 381 498-7277 / -1187251
Email: peter.danielis@uni-rostock.de

***Abstract*—The distributed computing of data is a challenging task in terms of the self-organizing task distribution and computing, especially if distributed computing systems are becoming very large and complex. Therefore, the distributed hash table (DHT)-based P2P system called DuDE has been developed to compute statistics of access nodes of Internet service providers in an efficient way. DuDE exploits the high failure resilience and scalability features of the DHT network Kad to achieve a high-performance distributed system, which avoids the bottlenecks of a centralized computing solution. To ensure highly available data, Reed-Solomon codes for reliable distributed data storage are utilized. For implementing DuDE, usual working steps of distributed computing have been extended to realize a highly scalable computing system. We have developed a simulation model for a large-scale DuDE network consisting of up to 9,000 access nodes for computing statistics. In this paper, simulation results are presented, which demonstrate that DuDE is able to almost linearly accelerate the distributed computing compared to a centralized solution while introducing low traffic overhead.**

Keywords*—P2P, Kad, Distributed Computing, Distributed Storing, Simulation, Scalability.

I. Introduction

In order to improve quality of service, to identify weak points in their networks, and to detect attacks on network components, Internet service providers (ISPs) need to compute statistics from log data. This log data is recorded by the access nodes (ANs) of an ISP's access network. The processing of ever increasing log data volumes for computing these statistics poses a significant challenge to ISPs [1]. Existing centralized computation systems will soon no longer be able to process these data volumes [1]. Moreover, solely a standard set of statistics can be generated. This is not even possible for all ANs of an ISP's access network but only for a single AN. Finally, the computation of long term statistics (LTS) covering several days of statistics is not supported at all.

To overcome these drawbacks, DuDE—a Distributed Computing System using a Decentralized P2P Environment—has been developed [2]. To network ANs in a self-organizing way, the distributed hash table (DHT)-based P2P network Kad is utilized. Thereby, an additional centralized statistics computation system can be saved. An average AN like, e.g., Freescale Semiconductor's PowerQUICC II Pro has 40 % of free computing resources available, which DuDE can use. Of its 1 GB RAM capacity, 400 MB are available [3]. To avoid the overloading of a single node, each AN provides that computing power to the statistics calculation, which it has currently available. Contrary to a centralized solution, the DuDE system therefore shows high scalability when log data volumes are increasing. The intrinsic high failure resilience and scalability features of the Kad network are combined with sophisticated redundant data storage by means of erasure resilient codes (ERCs), particularly Reed-Solomon codes, to achieve highly available data [4]. DuDE allows for complex statistics and LTS calculation for a single AN or even for all ANs. Common statistics like the number of dropped packets, which DSL Access Multiplexers (DSLAMs) usually compile, are supported. Moreover, DuDE's modular design simplifies the incorporation of new statistics formats. DuDE has been successfully implemented as software prototype running on a few ANs.

However, it is hardly possible to build a test setup with several thousands of ANs [2]. Therefore, we have developed a simulation model to investigate DUDE's scalability in large-scale networks in terms of the time to complete the statistics calculation and the generated traffic overhead. The simulation model comprises a network with up to 9,000 ANs. The statistics computing exemplarily consists in the determination of RAM and CPU load and packet loss of all ANs. Before we present and evaluate our simulation results in Section V, we give a brief description of the DuDE system based on [2] to be able to understand the obtained results. Below, the following main contributions are briefly described:

- Determination of a simulation setup based on a realistic ISP network.
- Evaluation of simulation results regarding scalability and traffic overhead.

The remainder of this paper is organized as follows: The related work is explained in Section II. Section III gives a brief description of DuDE. The simulation setup is explained in Section IV. Section V evaluates simulation results before the paper concludes in Section VI.

II. Related Work

The work in [5] proposes an approach for statistics calculation for a dynamic wide-area environment with heterogeneous nodes. It does not use a DHT but a semi-P2P structure for achieving good load-balancing and avoiding bottlenecks regarding the performance. Still, a detailed performance evaluation is missing and the computation of LTS is not described.

978-1-4673-7194-0/15/$31.00 ©2015 IEEE

Moreover, resource consumption as well as task and data distribution to achieve highly available data is not specified. DuDE addresses these unsolved issues concerning implementation and clearly demonstrates performance benefits compared to centralized computation systems.

In [6], the authors focus on compensating departed peers by indirectly collecting data from other peers. Increased data availability is achieved by a random network coding, however a central server is required to collect data from all peers. This central server still represent a computation bottleneck. DuDE uses Reed-Solomon codes for guaranteeing highly available data as well as distributed statistics computation. Thus, computation bottlenecks are omitted.

A P2P-based distributed computing architecture called CoDiP2P is presented in [7]. It allows for sharing the computing resources of users. The work is focused on description and evaluation of a tree-based P2P protocol, which shows high complexity for maintaining and restructuring the network. In contrast, our paper addresses the performance evaluation of a P2P-based distributed computing system rather than examining the deployed P2P protocol itself. The utilized protocol Kad is well-proven in practice. It is based on a flexible routing table and has significant advantages over tree-based protocols in terms of complexity and maintenance.

The CompuP2P framework enables distributed computing utilizing the DHT-based protocol Chord [8]. On top of the Chord protocol, further over-overlays including a resource trading and a service layer are provided. Contrary to CompuP2P, DuDE renounces the utilization of further over-overlays because of their overhead and solely uses a slightly extended version of the Kad protocol.

The project "JNGI" introduces a framework based on the hybrid P2P network JXTA for large-scale computations [9]. JXTA suggests three peer groups (monitor, task dispatcher, and worker group). Thereby, the assignment of peers to groups is not described. In DuDE, the peers are grouped on the basis of their available resources to achieve resource awareness. In the suggested framework, the task dispatcher polls the task dispatcher directly. In contrast, DuDE's job scheduler defines timeouts for workers for sending completed results and thus enforces rapid job completion. Moreover, JXTA may evoke inconsistency decreasing framework performance whereby Kad guarantees logarithmic lookup performance [4].

The works in [10] and [11] address peer statistics collection implemented on an additional over-overlay on top of a DHT-based P2P network. The over-overlay comprises functionality for information collection (e.g., loads and capacities) about the peers of the underlying DHT network. Contrary, DuDE's resource collection procedure to determine peers, which are suitable for computing tasks is directly implemented in the Kad network. Thereby, no additional overlay on the DHT network is necessary making DuDE a compact solution.

In [12], a hybrid network is presented, which comprises mobile nodes with high churn and nodes with higher reliability and performance. Nodes with high reliability are connected by the DHT-based P2P network Chord and ordinary mobile nodes with high churn are connected to them. This system is advantageous if a strongly heterogeneous network is given. DuDE entirely comprises reliable ANs with almost no churn, which corresponds to the Chord-based core network in [12]. However, Chord is inferior to Kad used for the DuDE system regarding its search complexity and flexibility [4].

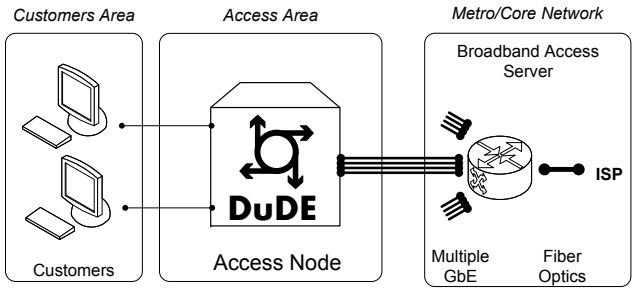

Fig. 1. Access network with DuDE node.

A hybrid approach is presented in [13] as well. A Chord-based routing is applied mainly for consistent data management with high churn. Although the hybrid approach reduces the problem of imbalanced distribution of the data through the formation of balanced clusters for certain data sets, the administrative effort becomes much more complex. Contrary, DuDE shows a better performance by applying Kad rather than Chord and utilizes ERCs rather than simple data replication. By using ERCs, not only less memory on the nodes is required but bandwidth is saved since less data needs to be transmitted. Currently, all data is considered to be of the same importance and therefore treated in the same way by ERCs. If DuDE will be used for other distributed computing use cases propectively, replication techniques based on data popularity from [14] could be used to achieve an even more increased DuDE efficiency.

III. DuDE Basics

P2P technology in access networks: Figure 1 shows a typical access network, where so-called ANs like IP DSLAMs build the access networks. The functionality of DuDE is directly implemented on these ANs. The ANs must cooperate to combine their computing power. Thereby, we avoid the usage of the client-server model as the server represents a bottleneck and single point of failure in the access network. Consequently, a decentralized system realized by exploiting the high scalability of P2P technology has been chosen. DHT-based systems as structured decentralized P2P systems offer the best trade-off between lookup and storage complexity [4]. DHT systems associate nodes and functions or data with the help of a hash function. Moreover, DHTs are self-organizing and do not need a central control instance. Additionally, failing peers can be compensated and detected automatically within the system by means of maintenance mechanisms. In addition to the good complexity of DHTs, they show a deterministic lookup thereby avoiding false negatives. Typical representatives of DHT-based P2P protocols are Chord [15], Tapestry [16], Pastry [17], and Kademlia [18]. The Kademlia protocol has been selected due to its best trade-off in terms of lookup and storage, flexible routing table, and its simple worst case analysis [4]. Every node has its own routing table containing some other nodes in the DHT network. This allows to perform lookups with the complexity of $log_2(N)$, whereby N denotes the number of nodes in the network.

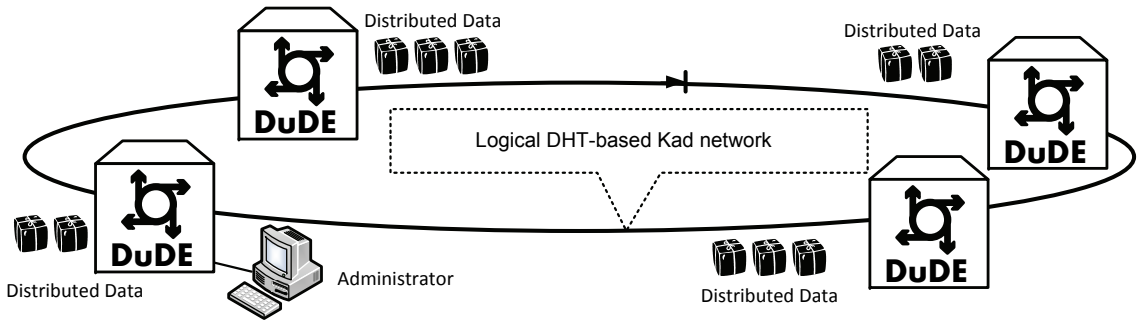

Fig. 2. Kad-based DHT ring connecting DuDE nodes for distributed statistics computing and distributed data storage.

Kad protocol modifications: Kad as an implemented realization of the Kademlia protocol has been used to realize DuDE [19]. Other P2P protocols such as Chord or Pastry could be adapted and used as well. However, especially Kad's high lookup performance makes it the best suitable choice for realizing DuDE [20]. All ANs are organized into a Kad-based DHT, whereby each DuDE node in the P2P system takes over the responsibility for log data part. DuDE stores data in the network redundantly, which increases data availability. Additional search objects have been implemented to extend the Kad protocol and to support distributed data collection. Moreover, new packets have been implemented to achieve an efficient communication between DuDE nodes. In Figure 2, the network is depicted as logical ring on top of the real network.

Performing the distributed computing: Selected distributed computing aspects are used to avoid overloaded components, which exist in centralized computation systems. Eligible ANs participating in the statistics computation may become task watcher and/or job scheduler. Their suitability depends on their available resources. DuDE nodes can themselves decide about their participation, which leads to a high grade of flexibility and load balance. A job scheduler is a high-performance node, which is responsible for distributing statistics computation parts (tasks) to ANs with sufficient available resources, which are called task watchers. Each task watcher computes a statistics part and sends it back to the job scheduler after complete computation. An ISP administrator is able to request computed statistics from any AN and does not need to be connected to the job scheduler.

In summary, the novel system DuDE combines P2P technology, an extended Kad protocol version enabling complex distributed data storage, and distributed computing.

A. Distributed file sets and LTS

DuDE nodes have a ring buffer available to be able to store log data represented by file sets. From this log data, statistics are calculated. However, this ring buffer has a limited storage capacity and is only able to store several hours of log data. For allowing to keep track of a special kind of statistics, LTS, which comprise statistics of a substantial time period of several day, ANs calculate LTS periodically and store it in an additional memory. By providing LTS, a more detailed analysis of each AN is enabled for ISPs. For the creation of LTS, DuDE possesses a component denoted as periodic accumulator (PA). Iteratively, the PA computes LTS from the log data in the ring buffer. New LTS are appended to existing data in the file set holding the LTS. Both file sets containing log data and LTS are encoded with erasure resilient codes (ERCs), which are detailed in Section III-D. Thereby, the PA is responsible for encoding LTS and another component called data spreader (DS) takes over the task of encoding the file sets holding the log data. After having divided the data into chunks by means of the ERCs, the chunks are distributed by PA and DS to the Kad network. In doing so, log data and LTS of each AN are available in the Kad network even if an AN should fail. As described in [2], PA and DS are designed in such a way to make sure that LTS and log data are distributed in the Kad network in good time before the ring buffer is full and old data is overwritten.

B. Global peer discovery

Because DuDE does not include a central instance, the peer number in the network is not known in advance. Still, for global statistics computation, all log data of nodes represented by file sets is needed. To achieve this, the name of the file sets containing the node name has to contain the unique AN ID, e.g., assigned by an administrative instance. Consequently, a search for file sets of nodes is also a search for nodes resulting from the association between node ID and filenames. Therefore, each filename of the file sets is unique and used for solely one node. The nodes' hash values are iteratively created by using a known fixed string like "Node" and an integer value, which is incremented for each node. It should be noted that even though ANs have unique IDs, hash values are assigned to them to establish a relation between data and nodes. The Kademlia Discovery (KaDis) algorithm discovers all nodes with a high probability regarding the given parameters. A detailed description and example can be found in [2].

C. Extended working process

As exemplification, 4 DuDE nodes with free resources (processor (CPU) and memory (MEM) resources) are assumed. Table I shows the resources of the 4 ANs, which can be computer performance measures like million instructions per second (MIPS) and floating point operations per second (FLOPS). The working process is presented based on this scenario. In accordance with Ian Foster's design process for designing a parallel algorithm [21], the DuDE process is structured into seven steps (see Figure 3). It is extended as DuDE needs to determine a job scheduler and task watchers resulting into an increased managing effort. However, thereby DuDE can work without a central instance.

AN	CPU	MEM	CPU+MEM
A	10	10	20
B	90	90	180
C	50	45	95
D	47	24	71

TABLE I. EXAMPLE SCENARIO WITH FOUR ANS. FREE CPU AND MEMORY RESOURCES ARE SHOWN.

Step 1: In step 1, DuDE node A performs a resource acquisition for job scheduler determination. The contacted nodes must respond within a specified time to become job scheduler. If they exceed this time they are not accepted as job scheduler and get a message to return to their initial state.

Step 2: As node B has most free resources available in this example, it becomes job scheduler. The other nodes get a message to return to their initial state so that they are not blocked anymore and can be requested from other nodes.

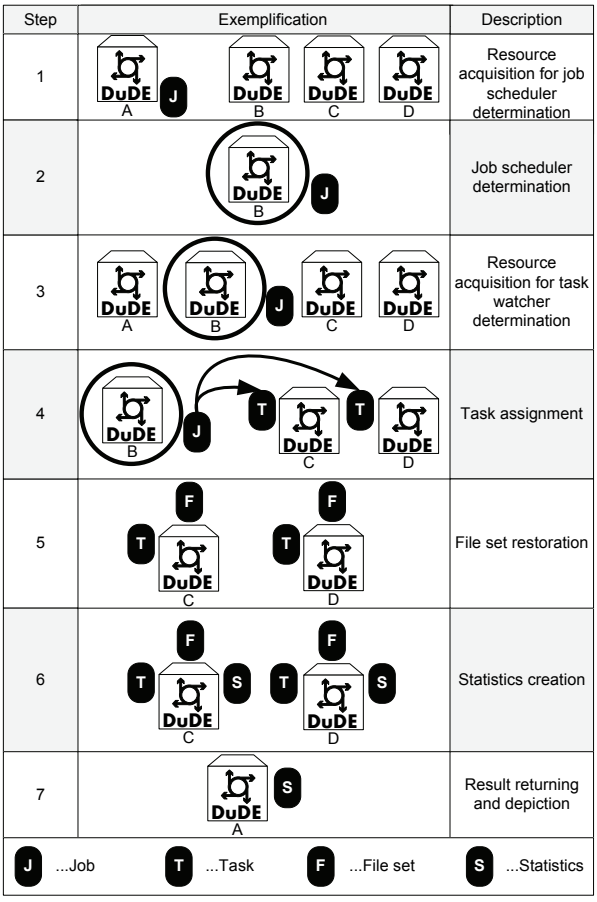

Fig. 3. DuDE's extended working process comprising seven steps.

Step 3: In the next step, node B (job scheduler) performs a resource acquisition again aiming at determining task watchers. As available resources of the requested nodes may have changed since step 1, it is necessary to perform a resource acquisition once more.

Step 4: In step 4, the job (J) is divided into single tasks (T) assigned to the selected ANs. In this example, node C becomes responsible for the first task and node D is instructed to execute the second task. By assigning a task to the most powerful node, tasks can be prioritized. Moreover, computation-intensive tasks should be given to nodes with sufficient resources to execute this task. When the job scheduler performs the resource acquisition, the requested nodes can either send their resources or block the request by sending a negative answer. They send their free resources if these are sufficient compared to the individually declared threshold for the task. ANs, which are not required, are freed to make sure that they are available again (e.g., node A in this example).

Step 5: In step 5, the task watchers restore the file sets by collecting the distributed data chunks from other ANs. If there are missing chunks, all nodes responsible for the data chunks are asked.

Step 6: From the file sets, the statistics are computed.

Step 7: The last step is used to return created statistics to the node that received the job. Therefore, the task watcher transmits its statistics to node B, which is the job scheduler. Node B is not directly connected to the administrator. Hence, statistics are transmitted to node A that received the job from the administrator. Node A finally depicts the results. The extended working process helps to structure the workflow for the implementation of a P2P-based distributed computing systems. Furthermore, the process achieves low communication overhead between nodes to guarantee a frictionless workflow.

D. DuDE software realization

Each DuDE node stores data chunks depending on its responsibility given by the DHT. Each node is assigned a unique ID called node ID. The node ID is represented by a hash value generated with the MD5 hash function [22]. To achieve high resilience against failing nodes, the file set is separated into n parts with a redundant share by the ERC algorithm. The Reed-Solomon codes [23] were chosen to generate the data chunks. m of n data chunks contain unchanged log data written in file sets. k of n data chunks contain coded information. Each coded data chunk can help to restore a file set if any other data chunk is lost. Whenever statistics have to be computed, distributed data (data chunks) are requested to regenerate the file set of a node containing the node's log data. Promising results from a prototype setup with seven DuDE nodes have been achieved [2], which are confirmed by new simulation results for a large-scale network consisting of thousands of nodes in this paper.

IV. SIMULATION SETUP

By means of simulation, the performance and generated traffic overhead of DuDE have been investigated. To show the behavior of thousands of nodes, a simulation model based on the functionality of the working prototype of DuDE has been developed [2]. A discrete time-driven C# simulator based on Microsoft's .NET framework with a time resolution of one millisecond has been developed to enable the fast simulation with several thousands of nodes. To accurately simulate DuDE, the .NET framework offers a comprehensive class library and a runtime environment called Common Language Runtime (CLR), which is responsible for memory management and the type system. The simulation is carried out stepwise. With every simulation step, all peers' routing tables are updated and the messages to be exchanged between peers approach their

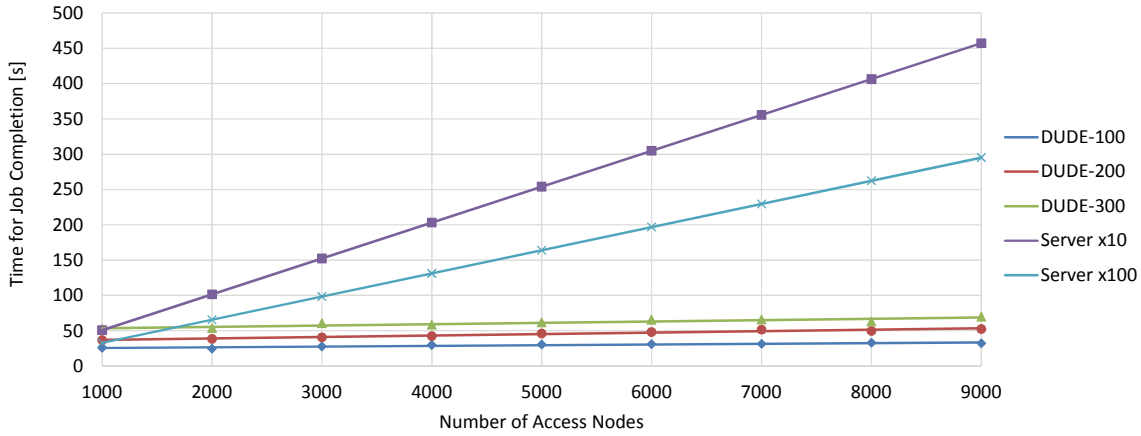

Fig. 4. Time for job completion. The DuDE system is compared to a centralized solution in the form of a server.

target. In accordance to practice, the ANs are networked as intermeshed star topology in the simulation. For the simulation of the network size, we oriented towards the backbone network of Deutsche Telekom, which consists of about 8,000 main distribution frames containing the DSL nodes, i.e., the ANs [24]. Particularly, in the simulation networks with increasing AN numbers ranging from 1,000 to 9,000 have been covered. All ANs are connected with each other with a bandwidth of 10 Gbit/s. The routing in the network considers the physical behavior of the switches and routers in the simulations. Therefore, delays can occur if the input buffer of a network element is filled faster than the data can be transmitted further.

A. Modeling of file sets for statistics calculation

A data type (RAM utilization, CPU load, or packet loss of an AN) is assigned to all file sets. The file set name is created from that data type and the AN ID, The size of a file set is determined by a random number generator whereby the random variable is uniformly distributed. The minimum size amounts to 1.5 MB and the maximum size equals 3 MB. Each file set is divided into 15 chunks (12 data chunks and 3 coded chunks) as described above, distributed to other ANs, and can be reassembled from the distributed chunks.

B. Modeling of jobs, tasks, and AN resources

The job consists of tasks for computing the RAM and CPU load and packet loss for all ANs in the network. Each task computes statistics for a maximum of 100 (denoted as *DUDE-100*), 200 (*DUDE-200*), or 300 (*DUDE-300*) ANs. For example, if a network comprises 8,000 ANs and each task calculates statistics for 300 ANs then 8000*3 (RAM, CPU, packet loss) / 300 = 80 tasks are necessary. Once a task watcher has collected all necessary file sets to execute its task, the calculation starts. An AN has a certain computing capacity available. This capacity specifies how many calculation steps an AN can execute at a point in time and is set to 30,000 calculation steps per second. The number of calculation steps necessary to process a file set during a task amounts to 2,000. This way of modeling the calculation steps corresponds to the computer performance MIPS and FLOPS respectively. It should be noted that the chosen values for the calculation steps represent estimates. However, even if the estimates deviate from practical values, the basic conclusions to be drawn from the simulations results remain the same. The DuDE system would solely outperform the centralized solution at an earlier or later point in time. For the comparison with a centralized solution in the form of a server, the computing capacity of the server is assumed to be ten times as high as that of an average AN (denoted as *Server x10*). To demonstrate the performance of DuDE, even a comparison with another server is included. This server has a computing capacity that is a hundred times as high as that of an average AN (*Server x100*).

V. Evaluation of the Results

The simulations comprise ISP networks with 1,000 up to 9,000 ANs, whereby their number has been increased in steps of 1,000. Following, the mean values of the time necessary for job completion and traffic overhead will be presented. 10 simulations per simulation run have been carried out.

High scalability regarding the time for job completion: As apparent from Figure 4, the DuDE system scales much better than a centralized solution in the form of a server. The time for job completion slightly linearly increases, i.e., it remains almost constant and thereby achieves a nearly perfect linear speedup. It can also be seen that the DuDE system becomes faster if one task comprises less statistics calculations, i.e., a task watcher calculates statistics for only a few ANs. This results from the fact that the more tasks a job is split into, the less is the processing load for each task watcher. If the DuDE-100 system is compared to a server that has a computing capacity ten times as high as that of an average AN, in case of 9,000 ANs the time is reduced by 95 %.

Low traffic overhead: In Figure 5, the relative traffic overhead per AN on average is depicted, which is introduced by the DuDE system. It is defined by dividing the overall amount of data in the network by the time needed for job completion and number of ANs. As apparent, the traffic overhead reduces with an increasing number of ANs. This results from the fact that the amount of data to be processed remains relatively constant per AN even though the absolute amount of data increases. At the same time, the time for job completion increases as the job scheduler has to distribute the tasks to more ANs and collect the results of statistics computation from more ANs successively. Depending on the number of statistic

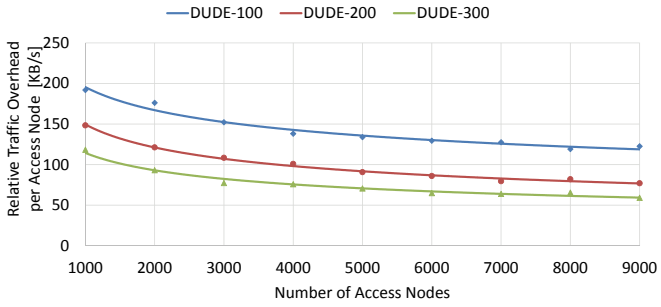

Fig. 5. Relative traffic overhead on average per AN introduced by DuDE.

calculations per task (*DUDE-100*, *DUDE-200*, or *DUDE-300*), the increase is different. For *DUDE-300*, the highest increase of job completion time is apparent resulting from fact, that each task watcher has to calculate statistics for 300 ANs, i.e., for more ANs than in the *DUDE-200* and *DUDE-100* case. Furthermore, the time for job completion rises with increasing AN number since delays due to high buffer fill levels in the switches and routers during the routing of the data occur. This results in the power function of the traffic overhead with increasing number of ANs. It should be noted that the traffic overhead per AN ranges between 59 and 192 KB/s. This is considered as low value as, e.g., a video on demand service (full HD) requires 6 MB/s per subscriber on average. It should further be noted that the largest share of traffic overhead is generated by packets, which are used to distribute data and to collect the distributed data. Since even the centralized solution would have to collect data, strictly speaking, only the part of distributing data should be considered as traffic overhead.

VI. Conclusion

DuDE is a recently developed system for the distributed computation of statistics based on P2P technology. DuDE utilizes a developed seven step process for distributed computing, which can be used as a possible approach for creating future distributed computing systems. Furthermore, Reed-Solomon codes are used for distributed data storage to achieve highly available data. To characterize DuDE in terms of scalability and traffic overhead, a simulation model of the DuDE node functionality has been developed to enable the simulation of a large-scale network. An extensive simulation with up to 9,000 ANs shows the high scalability in terms of the time to complete statistics computation. It is shown that the time a centralized solution would need can be reduced by up to 95 %. The generated traffic overhead solely occupies between 59 and 192 KB/s of the 10 Gbit/s connections of an AN and therefore does not impact the AN functionality. Summarized, DuDE realizes a high-performance distributed computing system with high availability and reliability of statistics in large-scale networks.

Prospectively, investigations on how to ensure a real-time behavior for statistics calculation in DuDE will be carried out.

Acknowledgement

This work is partly granted by the Research Fund Mecklenburg-West Pomerania, Germany, as well as the European Social Fund.

References

[1] Traffic Statistics by De-CIX. [Online]. Available: https://www.de-cix.net/content/network/Traffic-Statistics.html

[2] J. Skodzik, P. Danielis, V. Altmann, J. Rohrbeck, D. Timmermann, T. Bahls, and D. Duchow, "Dude: A distributed computing system using a decentralized p2p environment," in *IEEE 36th Conference on Local Computer Networks (LCN)*, Oct 2011, pp. 1048–1055.

[3] M. Ninnemann, "Freescale semiconductor powerquicc ii pro - performance and utilization," Broadband Access Division. Former Nokia Siemens Networks GmbH & Co. KG, November 2011.

[4] R. Steinmetz and K. Wehrle, *Peer-to-Peer Systems and Applications*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2005.

[5] S. Wu and T. Du, "GlobalStat: A statistics service for diverse data collaboration and integration in grid." HPC Asia, 2005, pp. 594–599.

[6] D. Niu and B. Li, "Circumventing server bottlenecks: Indirect large-scale P2P data collection." ICDCS, 2008, pp. 61–68.

[7] D. Castella, I. Barri, J. Rius, F. Gine, F. Solsona, and F. Guirado, "CoDiP2P: A Peer-to-Peer Architecture for Sharing Computing Resources," in *Advances in Soft Computing*, vol. 50, 2008, pp. 293–303.

[8] R. Gupta, V. Sekhri, and A. K. Somani, "CompuP2P: An Architecture for Internet Computing Using Peer-to-Peer Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, pp. 1306–1320, 2006.

[9] J. Verbeke, N. Nadgir, G. Ruetsch, and I. Sharapov, "Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment," in *3rd International Workshop on Grid Computing*. Springer-Verlag, 2002, pp. 1–12.

[10] K. Graffi, A. Kovacevic, S. Xiao, and R. Steinmetz, "SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems." ICPADS, 2008, pp. 279–286.

[11] Z. Zhang, S.-M. Shi, and J. Zhu, "SOMO: Self-Organized Metadata Overlay for Resource Management in P2P DHT," *Lecture Notes in Computer Science*, vol. 2735, pp. 170–182, 2003.

[12] H.-M. Xu, Y.-J. Shi, Y.-L. Liu, F.-B. Gao, and T. Wan, "Integration of cloud computing and p2p: A future storage infrastructure," in *ICQR2MSE,*, June 2012, pp. 1489–1492.

[13] J. Paiva, J. Leitao, and L. Rodrigues, "Rollerchain: A dht for efficient replication," in *12th IEEE NCA*, Aug 2013, pp. 17–24.

[14] A. Vijendran and S. Thavamani, "Least recently used replica replacement technique in distributed computing network," in *ICICA*, March 2014, pp. 104–108.

[15] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *ACM SIGCOMM*, 2001, pp. 149–160.

[16] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz., "Tapestry: A Resilient Global-scale Overlay for Service Deployment," in *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, 2004.

[17] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *IFIP/ACM International Conference on Distributed Systems Platforms*, 2001, pp. 329–350.

[18] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric." IPTPS, 2002.

[19] R. Brunner, "A performance evaluation of the kad-protocol," Master's thesis, University of Mannheim, November 2006.

[20] D. Stutzbach and R. Rejaie, "Improving lookup performance over a widely-deployed dht," in *INFOCOM*, 2006, pp. 1–12.

[21] I. Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison Wesley, 1995.

[22] R. Rivest, "The MD5 Message-Digest Algorithm," April 1992. [Online]. Available: http://tools.ietf.org/html/rfc1321

[23] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields. In: Journal of the Society for Industrial and Applied Mathematics," in *SIAM journal on applied mathematics*, 1966, pp. 300–304.

[24] S. Schweda, "Federal Administrative Court Rejects Competitors' Claim for Access to Telekom's Dark Fibre," 2010. [Online]. Available: http://merlin.obs.coe.int/iris/2010/3/article14.en.html