DuDE: A Distributed Computing System using a Decentralized P2P Environment

Jan Skodzik, Peter Danielis, Vlado Altmann,
Jens Rohrbeck, Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany
Tel./Fax: +49 (381) 498-7257 / -1187251
Email: {jan.skodzik;peter.danielis}@uni-rostock.de

Thomas Bahls, Daniel Duchow
Nokia Siemens Networks GmbH & Co. KG
Broadband Access Division
17489 Greifswald, Germany
Tel: +49 (89) 5159-22771 / -18237
Email: {thomas.bahls;daniel.duchow}@nsn.com

***Abstract*—The P2P-based system for the distributed computing of statistics called DuDE is presented. High scalability and failure resilience features of P2P are exploited to achieve a high-performance distributed system, which avoids the bottlenecks of a centralized computing system. To ensure high data availability, a sophisticated algorithm for distributed data storage is integrated. Furthermore, an algorithm for global peer discovery is presented, which allows for finding all data assigned to peers without the need for a central instance. For the realization of DuDE, common working stages of distributed computing are extended to enable a highly scalable computing system based on P2P technology. Generated results from a test system show a nearly perfect linear speedup for distributed computing as well as high processor and memory relief compared to a centralized solution.**

***Keywords*-P2P, DHT, Kademlia, Distributed Computing, Distributed Storing.**

I. Introduction

Statistics are essential for Internet Service Providers (ISPs) to improve quality of service and to detect bottlenecks in their network and attacks on network components. The processing of log data for the computation of these statistics poses a significant challenge to ISPs. On the one hand, existing centralized computation systems are not feasible or even not able to process the ever increasing log data volumes [1]. On the other hand, solely a standard set of statistics can be generated for a single access node. Especially, the computation of long term statistics (LTS) is not supported at all.

To remedy these deficiencies, DuDE—a Distributed Computing System using a Decentralized P2P Environment—has been developed. Access nodes of an ISP's access network are connected by a self-organizing distributed hash table (DHT)-based P2P network superseding an additional centralized system for statistics computation. These access nodes have a certain available storage and computing capacity, which may be used at no extra costs, assuming some idle time or storage capacity being left in an access node. Each access node contributes computing power to the statistics computation depending on its currently available resources, which avoids the overloading of a single node. Thus, the system scales with increasing log data volumes in contrast to a centralized solution. High resilience, which is an intrinsic P2P feature [2], in combination with sophisticated redundant data storage, guarantees high data availability.

DuDE supports the computing of complex statistics and LTS—both for a single access node and for all nodes. Typical statistics in DSL Access Multiplexers (DSLAMs) like the number of dropped and received packets are supported. Due to DuDE's modular design, new statistics formats can easily be incorporated.

Briefly summarized, the main contributions of this paper as follows:

- An extended working process is presented as basis for implementing a distributed computing system.
- A backup system and the grouping of resources to control the workloads by choosing special node IDs are presented.
- An algorithm to discover all peers in DuDE without the need for a central instance is described.
- Results from a real test system are presented and compared to a centralized solution. Furthermore, the performance and resource usage are discussed.

The remainder of this paper is organized as follows: The related work is explained in Section II. Section III presents DuDE's general aspects. It includes the distributed data system, an algorithm for global peer discovery, and introduces an extended working process for distributed computing consisting of seven stages. The software realization of DuDE is described in Section IV. Section V explains a test scenario. Corresponding results are evaluated in Section VI before the paper concludes in Section VII.

II. Related Work

In [3], a grid statistics service for a wide-area dynamic, heterogeneous environment is proposed. It is not DHT-based but utilizes a semi-P2P structure to achieve good load-balancing and to avoid performance bottlenecks. However, a detailed evaluation of performance is lacking and the computation of long-term statistics does not seem to be supported. Moreover,

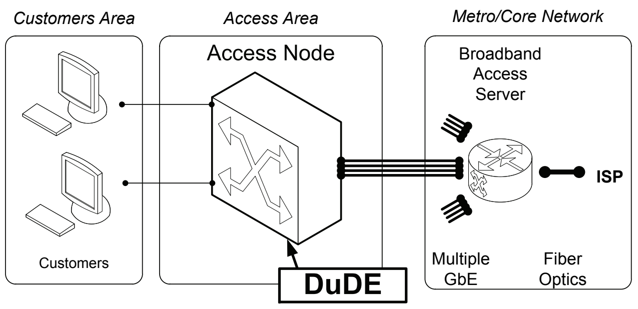

Fig. 1. Access network with DuDE node.

resource awareness does not seem to have been considered when distributing tasks and distributed data storage to achieve high data availability is not described. DuDE addresses these unsolved questions concerning implementation and clearly points out performance benefits compared to centralized computation systems.

The work proposed in [4] focuses on compensating departed peers by indirect data collection enabled by data distribution. Although random network coding is done to achieve increased data availability, still a centralized server has to collect data from all peers. The problem of the server as computation bottleneck is not solved but solely eased. DuDE both uses Reed-Solomon-Codes to ensure high data availability and distributed statistics computation thereby eliminating computation bottlenecks.

CoDiP2P is a P2P-based distributed computing architecture, which allows to share the computing resources of users [5]. The authors give a description and evaluation of their tree-based P2P protocol's functionality and properties. In contrast, this paper focuses on the evaluation of DuDE's performance for statistics computation rather than reviewing the deployed P2P protocol (Kad). Kad is well-proven in practice and has significant advantages over the tree-based protocol mainly regarding its complexity for restructuring and maintaining the network as Kad is based on a flexible routing table.

CompuP2P is a market economy-based framework for enabling Internet computing based on the DHT-based Chord protocol [6]. On top of the Chord protocol, further layers (over-overlays) such as the resource trading layer and the service layer are created. As opposed to this approach, DuDE deploys an extended version of the Kad protocol thereby renouncing the utilization of further over-overlays and their overhead.

The project "JNGI" is a framework for large-scale computations based on the hybrid P2P network JXTA [7]. JXTA's concept of peer groups is applied whereby three peer groups exist (monitor, task dispatcher, and worker group). Thereby, it is not clear whether peers are grouped on the basis of their available resources like done in DuDE to achieve resource awareness. As opposed to DuDE, the task dispatcher neither keeps track of which workers performing which tasks nor of job submitters but workers and the job submitter poll the task dispatcher directly. In contrast, DuDE's job scheduler enforces rapid job completion by defining time outs for workers and immediately sending completed results. As JNGI bases on JXTA, the system may suffer considerably from the resulting inconsistency decreasing its performance whereby Kad guarantees logarithmic performance [2].

The works in [8] and [9] focus on peer statistics collection by adding an over-overlay on top of a DHT-based P2P network. The over-overlay is responsible for collecting information (e.g., loads and capacities) of the peers organized in the underlying DHT network. Contrary, DuDE's resource collection procedure to determine adequate peers for computation tasks is directly implemented on the level of the DHT network. Thereby, no additional overlay on top of the DHT network is necessary.

III. DuDE in General

P2P technology in access networks: In Figure 1, a typical access network is depicted. Access networks comprise so-called access nodes like IP DSLAMs. DuDE is located on these access nodes. To achieve high computing power, access nodes have to cooperate and have to be organized. Thus, we renounce the usage of the client-server model as this paradigm is bottlenecked on the server. The solution is a decentralized system realized by exploiting P2P technology with its inherent scalability.

Thereby, a single point of failure like in the client-server model is prevented. DHT-based systems as structured decentralized P2P systems offer the best trade-off between search and storage complexity [2]. With the help of a hash function, the DHT creates an association between nodes and functions or data. DHTs are self-organizing and do not need a central control instance. Therefore, failing peers are compensated automatically. A lookup in DHTs is deterministic and no false negatives are possible. There is a wide range of DHT-based P2P protocols like Chord [10], Tapestry [11], Pastry [12], and Kademlia [13]. The Kademlia protocol has been chosen because of its routing table flexibility and high lookup performance. The routing table as central component of a DHT contains the contacts for the lookups.

Extended Kad protocol: DuDE uses Kad as an implemented realization of Kademlia protocol [14]. Thereby, all access nodes are organized into a Kad-based DHT. Each node in the P2P system is responsible for a part of log data. DuDE supports redundant data storage in the network. The Kad protocol has been modified and extended by means of additional search objects to support distributed data collection. Furthermore, new packets have been introduced to enable and improve the communication of DuDE nodes. Figure 2 depicts the network built as a logical Kad-based DHT ring on top of the real network topology.

Distributed computing: Selected aspects of distributed computing are applied to avoid overwhelmed components existent in centralized computation systems. Access nodes participating in the statistics computation can become job scheduler and/or task watcher depending on their available resources. A job scheduler is a very powerful node responsible for distributing parts of the statistics computation (tasks) to access nodes with sufficient available resources. Each task

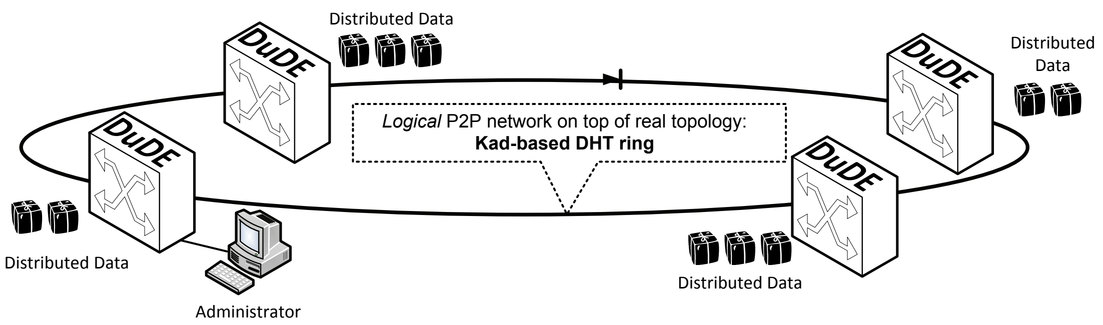

Fig. 2. Kad-based DHT ring for computing statistics and distributed data storage.

watcher computes its statistics part and returns it to the job scheduler after completion. An ISP administrator can request computed statistics from any access node and does not need to be connected to the job scheduler.

To summarize, the combination of P2P technology, an extended version of the Kad protocol enabling sophisticated distributed data storage, and distributed computing form the novel system called DuDE.

A. Distributed file sets and LTS

LTS comprise statistics of a substantial period. Thereby, network parameters can be observed over several days, which allows for a more significant analysis of the system behavior. DuDE nodes have a ring buffer as storing element for log data, which is the basis for LTS computation. However, the ring buffer's storage capacity is solely sufficient for storing several hours of data. Therefore, nodes have to periodically compute LTS and store it in an extra memory. The component, which creates the LTS, is called Periodic Accumulator (PA). The PA uses the actual log data of a node stored in the ring buffer and creates LTS in an iterative way. New relevant data is appended to already existing data in an LTS file. LTS are encoded with an erasure resilient code (ERC) algorithm. The ERC algorithm is explained in Section IV. Finally, created data is divided into chunks and stored in the P2P network. By using data compression for LTS, the stored data volume can be minimized. Furthermore, the distribution of the existing log data of the nodes is performed to keep data for creating complex statistics up-to-date. An element called Data Spreader (DS) encodes the file sets by means of the ERC algorithm as well and stores them in the P2P network in a distributed manner. Thereby, PA and DS have time parameters to coordinate the distribution of data. The minimum time interval for storing the data depends on the periodical creation and storage of the LTS. If a data set needs N steps to run through the ring buffer, LTS have to be created and distributed after step N-1 and before step N. However, if DuDE creates and distributes LTS before step N-1, redundant data distorts the LTS. If the creation and distribution of LTS happens after step N, DuDE misses data for the creation of LTS and statistics could be distorted as data is lost because the data is not stored in the ring buffer anymore. Time parameters of the PA are not bound to any conditions. However, tests with a real system have shown that the parameter should not be too small as this would cause a permanent update of the file sets. A permanent update causes a higher usage of resources for encoding with the ERC algorithm and additionally a high utilization of the communication channel. Too large selected time slots cause the creation of statistics based on too old data. Consequently, the size of the time slot has to be a trade-off between acceptable resource usage and actuality of data.

B. Global peer discovery

As DuDE does not have a central instance, the number of peers in the network is unknown. However, for global statistics computation, all data of nodes represented by file sets is needed. To achieve this, the name of the files containing the data of each node has to contain the node ID. Consequently, a search for file sets of nodes is also a search for nodes because of the association between node ID and filenames. Each filename of the file sets is unique and assigned to one node. The hash values of the nodes are created in an iterative way by using a known fixed string and an increasing integer value. It should be noted that even though access nodes have unique IDs, they are assigned hash values to establish a relation between nodes and data. The Kademlia Discovery (KaDis) algorithm shown in Figure 3 discovers all nodes with a high probability depending on the given parameters. In this example, the fixed string is "Node".

First, the node ID is calculated by creating the hash value and try to rebuild the data of a node. If it is successful the algorithm continues with the next node ID and resets the counter value K to zero. If not successful K is increased. If K equals the aborting threshold value A the algorithm stops. At this moment, the system decides that all possible data has been found. The chance of finding all data is depending on A. It is also possible to create statistics of nodes, which have left the P2P network because their data could still be distributed in the P2P network. In Table I, a demonstration scenario is presented illustrating the explained algorithm. A is set to three. The existing file sets of "Node1", "Node2" and "Node5" are stored in the P2P network. The file sets for "Node1" and "Node2" are found in the P2P network and successfully decoded by the ERC algorithm. After two non-successful decoding attempts for "Node3" and "Node4", K is increased to the value two,

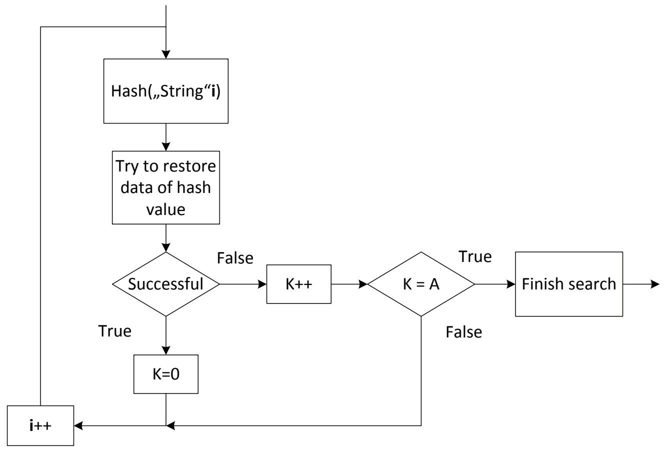

Fig. 3. KaDis algorithm for global peer discovery

which is still smaller than A. The ”Node5” file set is found and K is reset to zero. The KaDis algorithm searches for more possible file sets and stops after the file set of ”Node8”. Now, the algorithm assumes that there are no more file sets.

I	SUCCESSFUL	K	K >= A	FILE SET
1	yes	0	no	MD5(”Node1”).txt
2	yes	0	no	MD5(”Node2”).txt
3	no	1	no	MD5(”Node3”).txt
4	no	2	no	MD5(”Node4”).txt
5	yes	0	no	MD5(”Node5”).txt
6	no	1	no	MD5(”Node6”).txt
7	no	2	no	MD5(”Node7”).txt
8	no	3	yes	MD5(”Node8”).txt
Algorithm stops				

TABLE I
EXEMPLIFICATION OF KADIS ALGORITHM

C. The extended working process

As an exemplification, four DuDE nodes with free resources (processor and memory resources) were used. Table II shows the resources of the four nodes. Based on this scenario, the working process is presented.

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

TABLE II
EXAMPLE SCENARIO WITH FOUR NODES. FREE CPU AND MEMORY ARE SHOWN.

The process is structured into stages like in Ian Foster’s design process for designing a parallel algorithm [15]. The DuDE model has been extended to seven stages, which are depicted in Figure 4. It is extended because DuDE needs to declare a job scheduler and task watchers, which results into more managing effort. However, this approach allows DuDE to work without a central instance.

Stage 1: In the first stage, DuDE node A searches for possible resources to determine the job scheduler. Any node can become the job scheduler. The contacted nodes have to answer within a specified time slot to apply for the job scheduler. If they answer later they will not be accepted by the requesting node. These nodes get a message to get back to their initial state.

Stage 2: Node B has most free resources to offer and becomes job scheduler. All other nodes are freed at this moment. Freed nodes are nodes, which are not blocked anymore and are free for further requests from other nodes.

Stage 3: The job scheduler B searches for resources in stage three again. Like in stage one, there is another time slot, in which nodes can apply as task watchers. It is necessary to search for resources again to be able to react on fluctuations of DuDE nodes, which might have happened during stage 1 and 2.

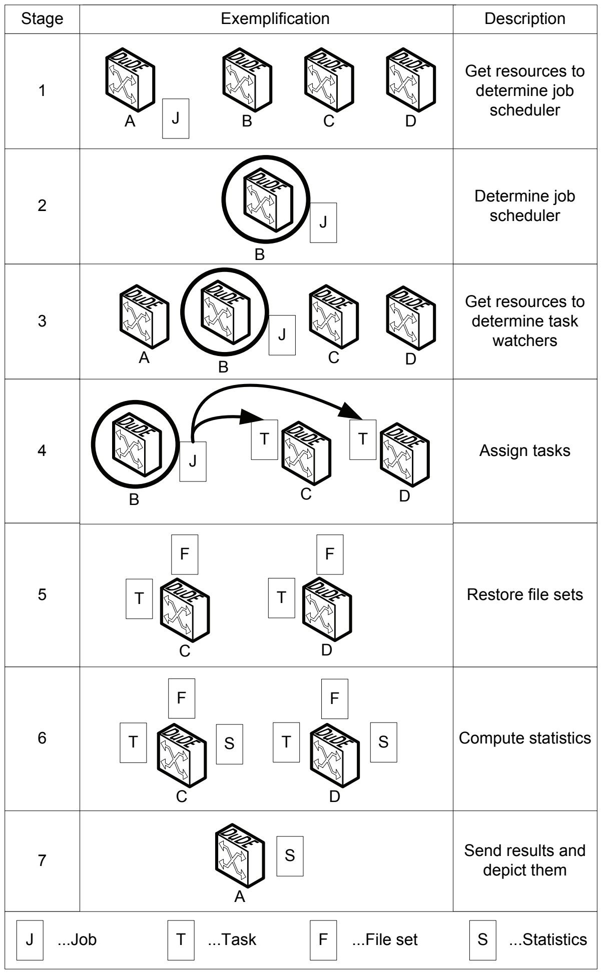

Fig. 4. Extended working process consisting of seven stages.

Stage 4: In stage four, the job is separated into single tasks. These tasks are assigned to chosen resources. In this example, node C gets the first task and node D gets the second task. It

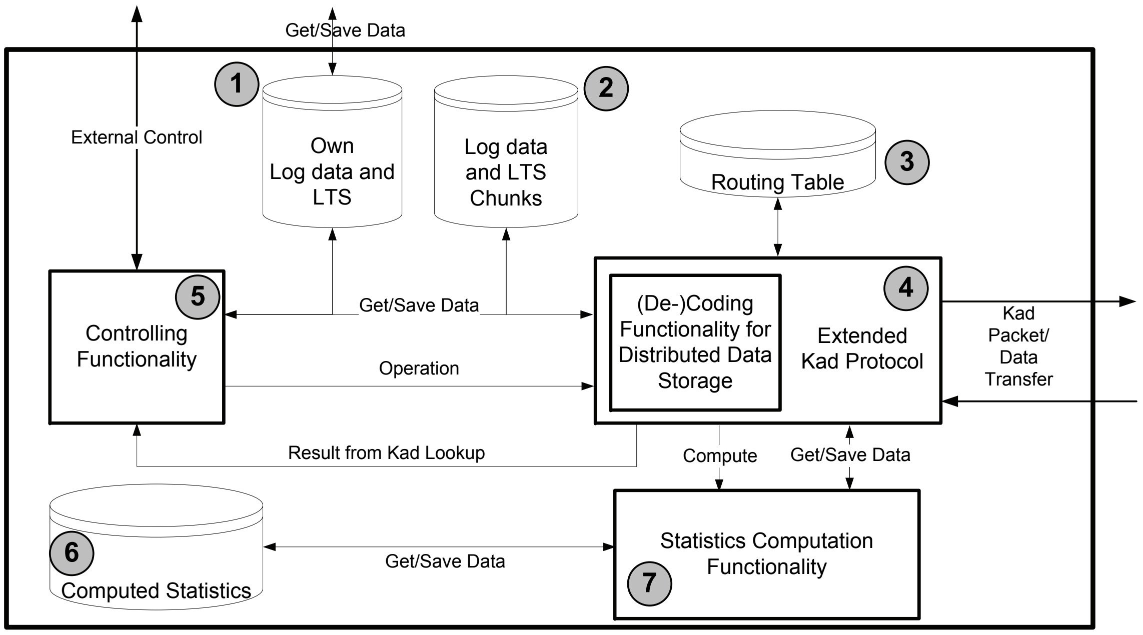

Fig. 5. System architecture of a DuDE node.

is an iterative approach to assign tasks to the most powerful node. This allows for the prioritization of tasks. Consuming tasks should be assigned to nodes with sufficient resources. There is a threshold for each node. When the job scheduler asks for the resources of nodes, they can decide either to send their resources or to block the request by sending a negative response. They send their free resources if the actual free resources are larger than the individually declared threshold. Resources, which are not needed, are freed to guarantee that requested nodes are available again. This applies to node A.

Stage 5: In stage five, the task watcher collects the data chunks to be decoded into according file sets. If there are missing files, all responsible nodes for the data chunks are asked.

Stage 6: Thereafter, the statistics are computed in stage six.

Stage 7: The last stage is used to send generated statistics to the node, which received the job. First, the task watcher sends its statistics to the job scheduler. Node B is not directly connected to the terminal. Therefore, statistics are sent to node A, which received the job from the administrator. Node A finally depicts the results. The presented extended working process helps to structure the workflow and serves as framework for the implementation of distributed computing systems based on P2P technology. Furthermore, the process offers an administrator the minimal necessary communication between nodes to guarantee a frictionless workflow.

D. Using a backup system

To achieve a higher reliability of the functional part of DuDE, a backup system has been integrated. The backup system is an optional feature and compensates failing task watchers. During the assignment of a task to a task watcher, a second node receives the task in parallel. Both nodes, the task watcher and the corresponding backup node, are working on the identical file sets to create the statistics. If the task watcher fails the backup node can immediately send the results. Therefore, the job scheduler has an internal time slot for receiving statistics. All statistics, which have not been sent to the job scheduler within the time slot, are requested from the corresponding backup node. The failing node gets a message to get back to the initial state.

E. Grouping of Resources

The working process is completely self-organizing. Moreover, the use of known fixed strings and an integer value allow for the grouping of resources. This grouping depends on a given parameter, which is part of the fixed string to generate the node ID. This parameter could contain geographic, ownership, or even capacity aspects. It is possible to push workloads to special groups of nodes by simply choosing predefined fixed strings. In addition, with the constrained choice of resources and grouping, DuDE allows the administrator to manage the workflow flexibly without any knowledge of the network.

IV. Software Realization

The system architecture of a DuDE node is depicted in Figure 5. The main components are a memory for storing a node's own log data and LTS (1), a memory with log data and LTS chunks (2), a routing table with contact information of other nodes (3), the extended Kad protocol (4) with integrated coding functionality for distributed data storage, and a block for receiving commands from an external control and initiating the computation of statistics (5). The log data chunks represent the file sets of the nodes when decoded. Furthermore, a memory used for storing demanded statistics (6) and the statistics computation functionality (7) including the PA and DS is contained. Data chunks are created by the coding functionality and distributed via Kad to achieve high data availability.

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 cryptographic hash function MD5 [16]. MD5 could generate more collision when using only several bytes of the generated hash value. However, this can be reduced by using more bytes or if needed choosing a larger hash value. Each byte can be presented as American Standard Code for Information Interchange (ASCII) characters. By using more bytes (characters) for the hash value, the probability of collisions is reduced. Another side effect is the not-printable characters. The node ID is part of the file names used in DuDE, which is not free of conflicts with Windows and Linux operating systems. Not all ASCII characters are printable, e.g., controlling characters for printers. Eliminating the not-printable character raises the probability of collisions. Therefore, the not-printable characters are replaced by their equivalent integer values.

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 [17] were chosen to generate the data chunks, which are stored in memory (2). 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 replace any uncoded data chunk of all M chunks. Whenever statistics have to be computed, distributed data (data chunks) are requested to regenerate the file set of a node.

V. Test scenario

To determine DuDE's performance and needed system resources, a test system consisting of seven identical PCs connected via one 10 Mbit D-Link switch has been set up.

Value	Description
File size	File size of the file sets
KaDis value A	Aborting value of the KaDis algorithm
File sets	Number of available file sets
Effort	Thousand calculation of Π with the BBP formula
Task	Number of tasks inside a job

TABLE III
Adjustable values for the test runs

Each PC has a 1.5 GHz Pentium 4 CPU and 512 MB RAM. The operating system is Windows XP SP3. The network setup has been chosen to demonstrate the speedup of DuDE in comparison to a centralized system, which is used today. Also, it is sufficient to show and validate DuDE's functionality in a scenario close to reality. The log data of a node is written into file sets. There are several variables, which have an impact on the results, e.g., the size of the data. Table III gives an overview about the different possible adjustable values. The file set size is set to 100 KB and the KaDis parameter A is set to one. A is one because the participants and file sets in the network are known and we do not need to look for further unknown nodes in the network. When using DuDE, the number of file sets equals the number of nodes in the P2P network. The number of file sets and task were varied.

The creation of statistics includes parsing of requested data in the file sets. Matched data is stored in a new result file, which represents the statistics. As an adequate way to achieve higher effort than simple parsing, the Bailey-Borwein-Plouffe (BBP) formula has been implemented exemplarily. If there is a match the BPP formula calculates the mathematical constant Π to the depth of nine shown in Formula 1 [18]. The effort parameter indicates thousand calculation steps. An effort with the value ten results into ten thousand calculations of Π to the depth of nine.

$$\pi = \sum_{k=0}^{9} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right) \quad (1)$$

There is no theoretical limit for the number of tasks of a given job. The system pushes each task to an extra resource. Six PCs become task watcher and one works as job scheduler. Therefore, the limitation is six tasks in this case. The next chosen variable is the number of available file sets in the system representing the data for creating the statistics of the nodes. With an increasing number of file sets, the nodes have to process more data, which directly results into a higher computing time. The main task of the system is to store the data of nodes and parse for searched values. To recover the data of a node, all parts of the node's file set must be collected and decoded.

During the tests, several aspects were investigated. Processor (CPU) and memory (MEM) usage indicate the utilized system resources of the PCs. Computing time represents the performance itself.

VI. Evaluation Results

The following measurements describe the characteristics of DuDE. The performance represented by the speedup and the scalability of DuDE is presented. Both DuDE and a centralized system with one node computed the same statistics based on same file sets.

Measured parameters of the nodes allow drawing conclusions about the behavior of the system with real workloads. The following parameters were measured for each test run:

- Processor and memory utilization
- The number of packets in the communication queue
- Time for finishing a job

The processor and memory utilization of all nodes are measured to analyze the utilized system resources of the PCs. Nodes computing statistics had a processor usage of 100% during the computation as each PC is solely equipped with a single core processor. On ANs DuDE processes are started with low priority. As a result, they won't have any impact on important system processes. The processor utilization applies to

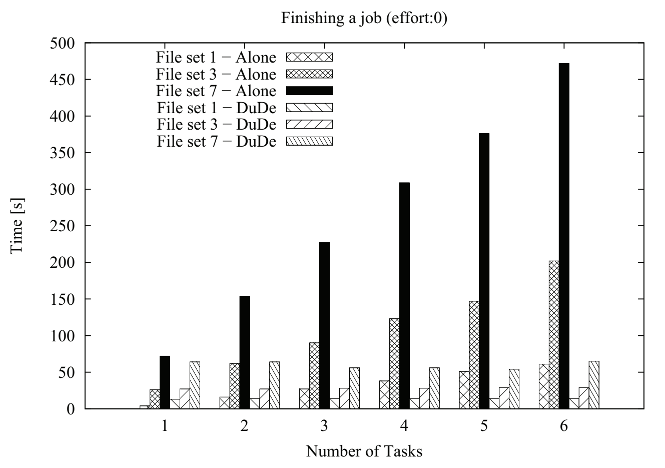

Fig. 6. Time required for processing a job for an increasing number of tasks with different number of file sets.

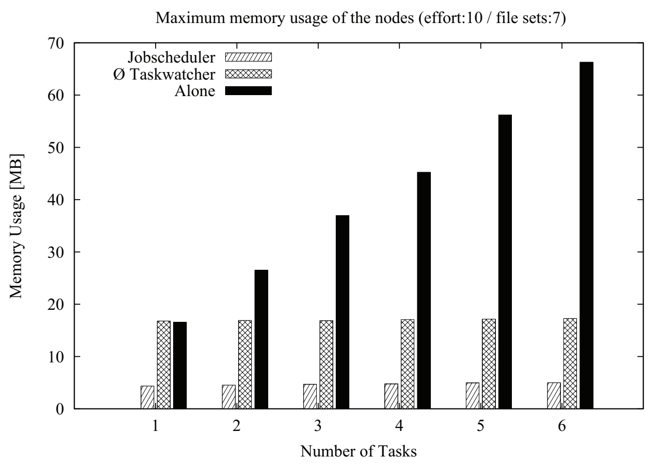

Fig. 8. Maximum memory usage of the nodes for an increasing number of tasks.

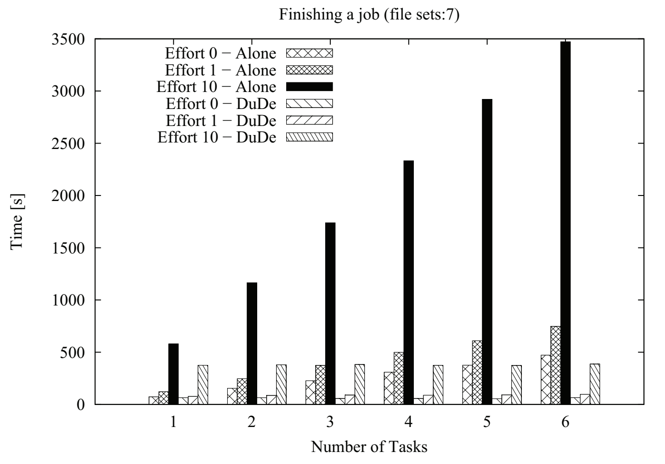

Fig. 7. Time required for processing a job with an increasing number of tasks at different effort.

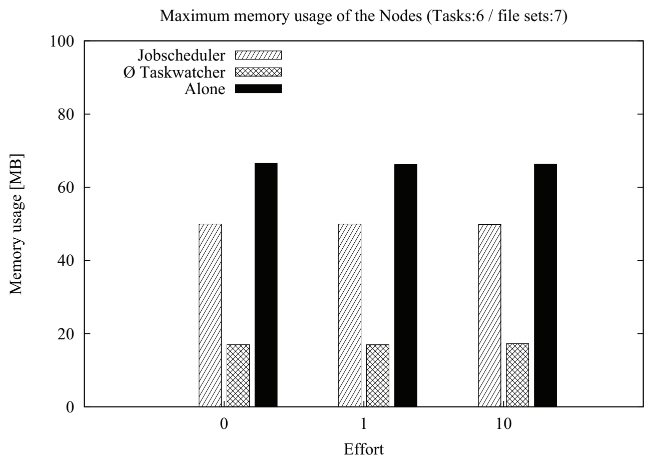

Fig. 9. Maximum memory usage of the nodes at different effort.

computing with DuDE and the single node. The job scheduler when using DuDE had a maximum processor utilization of 22%, which is a low load. However, the job scheduler only has to care about managing and observing the processing of the job. The communication between the nodes is done by UDP packets. The UDP packets are stored in a queue inside a DuDE node before they are processed. The number of packets in the queue raises with increasing communication traffic. Because of the low communication overhead and the immediate processing of the UDP packets, the number of packets in the queue never exceeded the value 1.

Moreover, Figure 6 shows that the needed time of a system with a single resource is increasing linearly depending on the number of tasks. The number of file sets equals the number of distributed data of nodes stored in the system. The computing time is nearly constant when using DuDE. The centralized system with one node needs more time than DuDE. The computing time is rising linearly with the number of tasks. Only with one file set and one task, the system with one single node is faster than the DuDE system. This is because DuDE needs more time to determine the job scheduler and the task watcher. This time is defined by the administrator. Furthermore, the needed computing time also depends on the number of file sets. Increasing the number of file sets implies the processing of more data.

The same behavior of the needed computing time is visible when varying the effort with the BBP formula for an increasing number of tasks. This behavior is shown in Figure 7. DuDE also needs more time for computing the statistics, because each task watcher must calculate Π with the BBP formula. As DuDE scales with an increasing number of tasks, the time until job completion is constant. I.e., if there are a sufficient number of task watchers, the computation time is independent of the number of tasks inside a job.

The maximum memory usage is the next investigated aspect. The number of tasks of a job is a main criterion for maximum memory usage. Figure 8 shows that the maximum memory usage for the job scheduler is constant as it only has the managing function. The constant maximum average memory

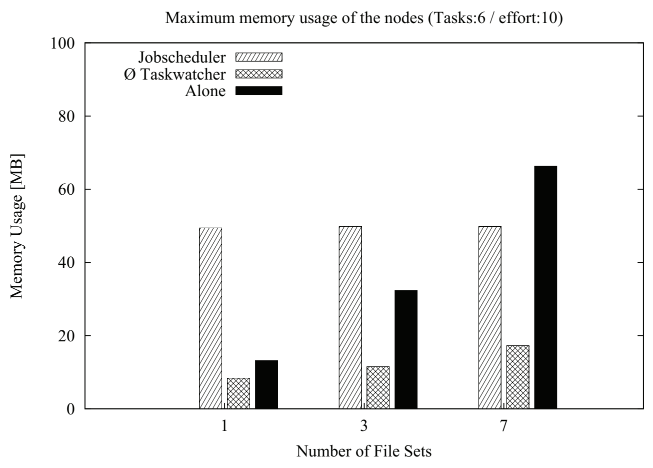

Fig. 10. Maximum memory usage of the nodes for an increasing number of file sets.

usage of the task watcher is due to the fact that each task inside a job is assigned to one task watcher. The single node instead has to process the whole data in the file sets alone, which is the reason for increasing maximum memory usage. However, Figure 9 depicts that the memory usage does not depend on the effort. A higher effort does not result in a higher data volume during computing the statistic for a complex statistic because the BBP formula is a pure mathematical calculation independent of any data. A higher data volume caused by an increasing number of file sets instead causes increased maximum memory usage. Figure 10 shows that the maximum memory usage of the task watcher and in the other case the single node is raising with an increasing number of file sets available in the system. The average maximum memory usage of all task watchers is lower than the maximum memory usage of the node running as a single resource. Also, the task watcher memory consumption increases much less. The job scheduler itself shows constant memory usage.

The achieved results clearly point out DuDE's benefits compared to a centralized computing system, which is used today. Actually, DuDE scales with raising number of DuDE nodes with nearly perfect linear speedup when the number of tasks is increased. The communication between the nodes does not have a significant impact on the results. The overhead of the communication is kept to a minimum in order to reach high performance.

VII. Conclusion

DuDE — a P2P-based system for the distributed computation of statistics — has been presented. An algorithm that discovers all file sets of the peers in the system without a central instance has been introduced. A seven stage process illustrated the working process and can be used as a possible approach for creating distributed computing systems based on P2P technology. Furthermore, Reed-Solomon-Codes are used for distributed data storage to achieve high data availability. A real test system and the corresponding results were compared with a centralized platform used today. The results show that DuDE's computing power scales with a raising number of DuDE nodes, which results in a nearly perfect linear speedup. Minimized communication between the nodes makes DuDE a highly scalable system concerning its computing power when the number of nodes increases. Not only system resources are saved but also a priorization of tasks and chosen resources is made possible. Future work will focus on exploring further use cases.

Acknowledgment

The authors would like to thank the Broadband Access Division of Nokia Siemens Networks GmbH & Co. KG in Greifswald, Germany for their inspiration and continued support in this project. This work is partly granted by Nokia Siemens Networks.

References

[1] Traffic Statistics by De-CIX. [Online]. Available: https://www.de-cix.net/content/network/Traffic-Statistics.html
[2] R. Steinmetz and K. Wehrle, *Peer-to-Peer Systems and Applications*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2005.
[3] S. Wu and T. Du, "GlobalStat: A statistics service for diverse data collaboration and integration in grid." HPC Asia, 2005, pp. 594–599.
[4] D. Niu and B. Li, "Circumventing server bottlenecks: Indirect large-scale P2P data collection." ICDCS, 2008, pp. 61–68.
[5] 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.
[6] 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.
[7] J. Verbeke, N. Nadgir, G. Ruetsch, and I. Sharapov, "Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment," in *In Proceedings of the 3rd International Workshop on Grid Computing*. Springer-Verlag, 2002, pp. 1–12.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric." IPTPS, 2002.
[14] R. Brunner, "A performance evaluation of the kad-protocol," Master's thesis, University of Mannheim, November 2006.
[15] I. Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison Wesley Pub Co Inc, 1995.
[16] R. Rivest, "The MD5 Message-Digest Algorithm," April 1992. [Online]. Available: http://tools.ietf.org/html/rfc1321
[17] 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.
[18] D. Bailey, P. Borwein, and S. Plouffe, "ON THE RAPID COMPUTATION OF VARIOUS POLYLOGARITHMIC CONSTANTS," in *Mathematics of Computation,*, vol. 66, no. 218, 1997, pp. 903–913.