Eley: On the Effectiveness of Burst Buffers for Big Data Processing in HPC systems

Orcun Yildiz, Amelie Chi Zhou, Shadi Ibrahim

Eley: On the Effectiveness of Burst Buffers for Big Data Processing in HPC systems

Orcun Yildiz, Amelie Chi Zhou, Shadi Ibrahim
Inria Rennes Bretagne-Atlantique, Rennes, France
{orcun.yildiz,chi.zhou,shadi.ibrahim}@inria.fr

***Abstract*—Burst Buffer is an effective solution for reducing the data transfer time and the I/O interference in HPC systems. Extending Burst Buffers (BBs) to handle Big Data applications is challenging because BBs must account for the large data inputs of Big Data applications and the performance guarantees of HPC applications – which are considered as first-class citizens in HPC systems. Existing BBs focus on only intermediate data of Big Data applications and incur a high performance degradation of both Big Data and HPC applications. We present *Eley*, a burst buffer solution that helps to accelerate the performance of Big Data applications while guaranteeing the performance of HPC applications. In order to improve the performance of Big Data applications, *Eley* employs a prefetching technique that fetches the input data of these applications to be stored close to computing nodes thus reducing the latency of reading data inputs. Moreover, Eley is equipped with a full delay operator to guarantee the performance of HPC applications – as they are running independently on a HPC system. The experimental results show the effectiveness of *Eley* in obtaining shorter execution time of Big Data applications (shorter map phase) while guaranteeing the performance of HPC applications.**

Keywords*—*HPC, MapReduce, Big Data, Parallel File Systems, Burst Buffers, Interference, Prefetch.

I. Introduction

With the arrival of the era of Big Data, we have witnessed the emergence of a new scalable data management and processing paradigm through the MapReduce model [1], [2] for data-intensive processing. MapReduce is adopted in both industry and academia due to its simplicity, transparent fault tolerance and scalability. For instance, Carnegie Mellon University is using MapReduce clusters for data analysis on several scientific domains including computational astrophysics, computational neurolinguistics, natural language processing and social networking analysis [3]. The success of MapReduce led to the emergence of new types of applications (e.g., stream data processing, graph processing, analysis of large scale simulation data) where obtaining timely and accurate responses is a must. However, commodity machines-based Big Data environments reach their limit due to being failure-prone and providing moderate performances.

Why Big Data processing on HPC Systems? High performance computing (HPC) systems are known for providing the maximum computing capability. They are equipped with high-speed networks, thousands of nodes with many cores and large memories [4], [5]. For instance, Sunway TaihuLight, No.1 in the top 500 supercomputers list, is a 10,649,600 processor supercomputer with a Linpack performance of 93 petaflop/s[1]. *Given the high performance nature of HPC systems and the increased need of the Big Data community for fast Big Data processing, the Big Data community is rapidly moving towards ways to leverage these HPC systems [6], [7].*

Big Data processing on HPC Systems: Challenges. Introducing Big Data processing to these systems is not trivial; it comes with its own set of challenges. First, one should be aware of the different architectural designs in current Big Data processing and HPC systems. Big Data processing clusters have shared-nothing architecture (e.g., nodes with individual disks attached) and thus they can co-locate the data and compute resources on the same machine (i.e., data locality) when performing the task scheduling. On the other hand, HPC clusters employ a shared architecture (e.g., parallel file systems) [8] which results in separation of data from the compute nodes. Big Data systems are mainly optimized to sustain high *data locality* which is the major contributing factor to the application performance [9], [10], [11]. However, the same approach on the data locality can not be applied in HPC systems since all the nodes have an equivalent distance to the input data. In addition, employing a shared parallel file system also results in remote data transfers through network when Big Data applications performing I/O thus resulting in a *higher latency*. Moreover, *I/O interference* (i.e., performance degradation observed by any single application/task running in contention with other applications/tasks on the same platform [12]) is a well-known performance bottleneck for HPC applications due to the large size and shared architecture of HPC clusters. Hence, *the impact of both latency and interference will be amplified, if HPC systems are to serve as the underlying infrastructure for Big Data processing.*

Burst Buffers for Big Data applications in HPC Systems: Current status. Many research efforts have been dedicated to overcome the high latency problem by introducing an intermediate layer — Burst Buffers (BBs) — with high throughput storage devices [13], [14], [15], [16] especially for storing the *intermediate data* (i.e., map output for batch jobs and temporary output produced between stages for iterative jobs). For example, Islam et al. [13] utilized NVRAM as an intermediate storage layer (i.e., burst buffer) between compute nodes and Lustre which improved the application performance by 24%. Wang et al. [16] leveraged SSDs for storing the intermediate data. Chaimov et al. [14] used a separate set of nodes (burst buffer nodes) with NVRAM as a storage space to achieve a better scalability by reducing the latency when reading/writing the intermediate data. Unfortunately, aforementioned works ignore the importance of input data despite the significant amount of work dedicated to improve the

[1] http://www.top500.org

application performance by sustaining high input data locality, as we mentioned previously. In addition, these work proposed the adoption of burst buffers in a similar way to the traditional use of burst buffers on HPC systems which aim to minimize the I/O time by absorbing the checkpointing data of scientific applications. In contrast, Big Data applications mostly run in batches therefore there is a continuous interaction with the parallel file system for reading the input data. Lastly, none of these efforts considered the interference problem which can contribute to a significant performance degradation — not only for Big Data applications but also for the first class HPC applications — by up to 2.5x [12]. Hence, *as we argue in this paper, current efforts and solutions to extend burst buffers for Big Data applications in HPC systems may fail in practice to achieve the desired performance and this may hinder the convergence of Big Data processing and HPC.*

Contributions. In an attempt to extend Burst Buffers for Big Data applications on HPC systems, in this paper, *we present Eley, a burst buffer solution that aims to accelerate the performance of Big Data applications while providing performance guarantees for HPC applications (i.e., meeting the completion time of each I/O phase as if HPC application is running alone).* A typical job of Big Data applications consists of several iterations (waves) depending on the total input size and the cluster capacity (i.e, the number of map tasks which can simultaneously run in the cluster). For example, if the cluster capacity is set to 100 map tasks where each map task processes one block of 128 MB, a job of 100GB data inputs will be executed in 8 waves. This is common for large scale scientific Big Data applications: during 9 months in a research cluster (i.e., M45) with 400 nodes [17], the amount of processed data was almost 900 TBs and almost 100 of executed jobs have an input data of 4.9 TBs. Thus, *Eley* employs a prefetcher technique that fetches data inputs of next iterations while computing nodes are still busy processing data inputs of previous one. This allows to have data inputs to be stored on a low-latency device close to computing nodes and therefore results in reducing the latency of reading data inputs of Big Data applications. However, data prefetching may come at a high cost for the HPC applications: the continuous interaction with the parallel file system (i.e., I/O read requests) may introduce a huge interference at the parallel file system level and thus end up with a degraded and unpredictable performance for HPC applications. To this end, we equip *Eley* with a full delay operator to guarantee the performance requirements of HPC applications. Full delay operator achieves this by delaying the prefetching to the end of I/O phase of HPC application. This allows HPC application to perform its I/O operations alone and thus avoids any interference. Our evaluations show that *Eley* reduces the execution time of Big Data applications by up to 30% compared to the naive burst buffer solution (denoted as *NaiveBB*) [13], [14], [15], [16] while guaranteeing the performance requirements of HPC applications. In summary this paper makes the following contributions:

- We propose *Eley*, a burst buffer solution for reducing the latency of reading data inputs for Big Data applications while guaranteeing the performance of HPC applications. The intuition behind *Eley* is that, on the one hand, Big Data applications are often executed in waves, thus, it is desirable to have data close to compute nodes once required. On the other hand, the heavy I/O requests introduced by Big Data applications to the parallel file system may adversely impact the performance of the (high priority) HPC applications, thus it is important to alleviate this impact as much as possible (Section II).
- The experimental results demonstrate the effectiveness of *Eley* in obtaining shorter execution time for Big Data applications (shorter map phase) while maintaining the performance of HPC applications. We believe that these results are encouraging and will help to bring forward the convergence between Big Data and HPC (Section III).

II. System Overview

In this section, we present an overview of the burst buffer solution (i.e., Eley) and describe our main design principles.

A. Burst Buffer Enabled HPC System

With the convergence between Big Data and HPC, many Big Data applications are moving to HPC systems to benefit from their high computation capabilities. In our design, Eley is located between compute nodes and parallel file system, where compute nodes only interact with the burst buffer nodes for all I/O phases of the Big Data application, including reading input data and reading/writing intermediate data. Thus, interference mainly occurs at the file system level due to incoming I/O requests of burst buffer nodes and HPC applications.

Different from the existing studies on burst buffer, in this paper, we instruct Eley to prefetch the input data of Big Data applications to optimize the I/O performance of those applications. In each prefetching operation, a subset of the input data (i.e., one wave) is prefeched for the next wave of computation. By doing so, we can overlap the I/O and computation phases of Big Data application to further hide the high latency of remote data accesses.

When we co-locate HPC and Big Data applications, prefetching operations may interfere with the I/O operations of HPC applications. Thus, we design a *prefetcher* component in the burst buffer to manage the prefetching of input data in a controlled manner. Specifically, our prefetcher component takes into consideration the performance requirements of HPC applications and employs a full delay operator if necessary. This operator delays the prefetching to the end of I/O phase of HPC application. This allows HPC application to perform its I/O operations alone and thus avoids any interference. Therefore, Eley can satisfy the performance requirements of HPC applications while improving the performance of Big Data applications.

B. Design Principles

We follow the following principles when designing the burst buffer enabled HPC system, which aim to address the challenges of running Big Data applications on HPC systems without introducing performance degradation to the HPC applications.

Enabling efficient Big Data processing on HPC systems: The separation of storage resources from the compute nodes

in HPC systems requires repetitive data transfer through the network, which results in a high I/O latency. In this paper, we focus on Big Data applications with huge input data sizes which are executed in multiple waves. Thus, the high I/O latency can severely degrade the performance of Big Data applications on HPC systems. We mitigate the latency problem by locating low-latency burst buffer devices (i.e., RAMDisk) close to the compute nodes to reduce the latency of reading data inputs of Big Data applications. Existing burst buffer studies mainly focus on mitigating the I/O latency on the intermediate data of Big Data applications. However, by analyzing traces collected from three different research clusters, we observe that the intermediate data size is smaller than 20% of the input data size for over 85% of the applications [17]. Thus, it is important to also mitigate the high I/O latency in reading the input data. To this end, we equip our burst buffer with a prefetcher component.

Guaranteeing performance requirements of HPC applications: Running Big Data applications in HPC systems should not introduce much performance degradation to the HPC applications, which usually involve important scientific simulations. Thus, one of our key design principles is to maintain the performance of HPC applications while improving the I/O performance of Big Data applications. To this end, we equip our prefetcher with a full delay operator to help satisfying our design objective.

III. Experimental Evaluation

In this section, we experimentally evaluate the effectiveness of Eley in improving the Big Data application performance while maintaining the performance guarantees of HPC applications. First, we present the methodology used to carry out our experiments. We then provide a detailed analysis of the evaluation results.

A. Methodology

1) Platform description: The experiments were carried out on the Grid'5000 testbed [18]. We used the Rennes site; more specifically we employed nodes belonging to the *parasilo* and *paravance* clusters. The nodes in these clusters are outfitted with two 8-core Intel Xeon 2.4 GHz CPUs and 128 GB of RAM. We leveraged the 10 Gbps Ethernet network that connects all nodes of these two clusters. Grid'5000 allows us to create an isolated environment in order to have full control over the experiments and obtained results.

2) System deployment: We used Spark version 1.6.1 to execute Big Data applications. We configured and deployed a Spark cluster using 33 nodes on the *paravance* cluster. One node consists of the Spark master, leaving 32 nodes to serve as slaves of Spark. We allocated 8 GB per node for Spark instance and set the Spark's default parallelism parameter (spark.default.parallelism) to 256 which refers to the number of RDD partitions. Each Spark slave has 16 map tasks thus Spark cluster can execute 512 map tasks in one iteration.

In our burst buffer design, we use low-latency storage devices as a storage space. To emulate this, we used Alluxio version 1.3.1, which exposes RAMDisk of the burst buffer nodes to the compute nodes as an in-memory file system. We configured and deployed 8 burst buffer nodes on the same cluster (*paravance*) as compute nodes to emulate that burst buffer nodes are closer to compute nodes compared to the parallel file system. Each burst buffer node provides approximately 32 GB of storage capacity. The OrangeFS file system (a branch of PVFS2 [19]) version 2.8.3 was deployed on 6 nodes of the *parasilo* cluster to serve I/O requests from both Big Data and HPC applications. We select 6 PVFS nodes using the same setting as existing studies [12].

3) Workloads: We selected two representative Big Data workloads including Sort and Wordcount, which are parts of the HiBench [20], a Big Data benchmarking suite. Wordcount is a map-heavy workload with a light reduce phase. On the other hand, Sort produces a large amount of intermediate data which leads to a heavy shuffling, therefore representing reduce-heavy workloads. For both workloads, we use 160GB of input data generated by RandomTextWriter of HiBench suite. Both workloads are executed with five iterations, where each iteration processes input data size of 32GB.

For HPC workloads, we use IOR [21] which is a popular I/O benchmarking tool for HPC systems. For each set of experiment, we run IOR side by side with the Big Data workloads using the same number of iterations. In each iteration, we use IOR to emulate the I/O requests of HPC applications. Between each request, IOR processes sleep for 20 seconds to emulate the computation time of HPC applications.

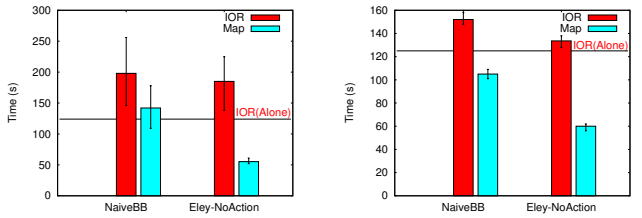
4) Comparisons: For our experiments, we adopt the following state-of-the-art burst buffer solutions as comparisons to *Eley*.

- *NaiveBB.* As in the existing burst buffer solutions [13], [14], [15], [16], *NaiveBB* uses burst buffer only for storing the intermediate data of Big Data applications. While it is already shown that this approach can improve the performance of Big Data applications, different from *Eley*, it does not consider the performance requirements of HPC applications and the I/O latency problem in the input phase.
- *Eley-NoAction.* This approach performs naive prefetching to copy the input data of Big Data from parallel file system to burst buffer nodes. Different from *Eley*, this approach is not aware of the performance requirements of HPC applications.

All solutions are evaluated using the same burst buffer storage space (i.e., RAMDisk) for fair comparison. We evaluate the completion time of both Big Data and HPC applications obtained by the three compared solutions to demonstrate the effectiveness of Eley. For Big Data applications, we only report the completion time of the map phase which is the main difference between Eley and the compared solutions.

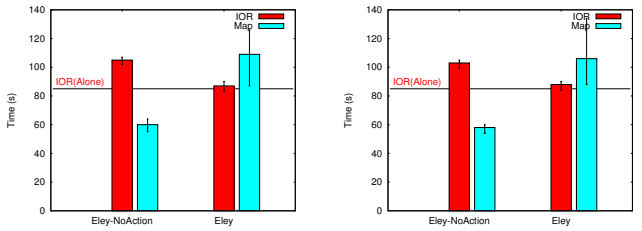
B. Key Results

We perform two sets of experiments. First, we evaluate the effectiveness of prefetching technique in improving the performance of Big Data applications, by comparing *Eley-NoAction* with *NaiveBB*. We then evaluate the effectiveness of the full delay operator of *Eley* in meeting the performance requirements of HPC applications by comparing *Eley* with *Eley-NoAction*.



(a) Sort (b) Wordcount

Fig. 1: Performance comparison between *Eley-NoAction* and *NaiveBB*. The horizontal line shows the IOR performance when it is running alone.



(a) Sort (b) Wordcount

Fig. 2: Performance comparison between *Eley* and *Eley-NoAction*. The horizontal line shows the IOR performance when it is running alone.

1) The effectiveness of prefetching: In this experiment, we execute IOR on a cluster of 32 nodes where one process per node issues a 4GB write request with chunk size of 1MB. Figure 1 demonstrates that using burst buffer for prefetching the input data of Big Data applications can greatly improve the performance of these applications. *Eley-NoAction* reduces the map time of Sort and Wordcount by 61% and 43%, respectively, compared to *NaiveBB* which reads input data from the parallel file system directly. It can be observed that the I/O time of the IOR workloads are also reduced when using prefetching for Big Data applications, by 7% and 12% when co-locating with Sort and Wordcount, respectively. This is mainly due to the fact that burst buffer aggregates the I/O requests sent from a large number of Spark nodes to a small set of burst buffer nodes and hence reduces the impact of interference on the performance of IOR workloads. For instance, while 32 compute nodes, with 16 Spark mappers each, request the I/O from PVFS in *NaiveBB*, only 8 concurrent fetcher threads on 8 burst buffer nodes perform prefetching with *Eley-NoAction*. This is inline with the existing studies in the literature [22], [23] which demonstrate that aggregation of I/O requests help to reduce the I/O interference. However, this small improvement will not be enough to meet the performance requirements of HPC applications, as highlighted in the horizontal line in Figure 1. For instance *NaiveBB* and *Eley-NoAction* leads to violation of the performance requirements for IOR workload up to 53% and 42%, respectively, Therefore, we next discuss the effectiveness of the full delay operator of *Eley*.

2) The effectiveness of Eley in guaranteeing the performance of HPC applications: In this experiment, we execute IOR on a cluster of 8 nodes, where one process per node issues a 8 GB write request with chunk size of 1MB. Figure 2 shows that *Eley* can meet the performance requirements of HPC applications thanks to the full delay operator. Full delay operator allows HPC application to perform its I/O operations alone and thus avoids any interference. On the other hand, *Eley-NoAction*, which is not aware of the performance requirements of HPC applications, leads to performance degradation of IOR workload by 21% and 17% (compared to the IOR performance when it is running alone as shown in the horizontal line in Figure 2) when it is co-located with Sort and Wordcount applications, respectively. It is also important to note that although prefetching with full delay operator leads to less performance improvement for Big Data applications, *Eley* still improves the performance of these applications by up to 30% compared to *NaiveBB* without degrading the performance of HPC applications.

IV. Discussion and Future Work

In this paper, we propose *Eley*, a burst buffer solution which takes into consideration both the input data and intermediate data of Big Data applications in HPC systems. Our goal is to accelerate the performance of Big Data applications without degrading the performance of HPC applications. To achieve this goal, *Eley* prefetches the input data of Big Data applications before the execution of each iteration. By prefetching, we are able to overlap the I/O and computation time to improve the performance of Big Data applications. However, data prefetching may introduce huge I/O interference to the HPC applications and thus end up with a degraded and unpredictable performance for HPC applications. To this end, we equip *Eley* with a full delay operator to guarantee the performance requirements of HPC applications. Our evaluation results show that *Eley* reduces the execution time of Big Data applications by up to 30% compared to the naive burst buffer solution [13], [14], [15], [16] while guaranteeing the performance requirements of HPC applications.

Thanks to these encouraging results, we plan to equip *Eley* with more operators (e.g., tuning the number of prefetcher threads), besides full delay. In particular, full delay can harm the performance of Big Data applications significantly when the HPC application has a long running phase. Therefore, these operators can help *Eley* in bringing higher performance improvements for Big Data applications while guaranteeing the performance of HPC applications. To achieve this, we plan to investigate interference and performance models for both Big Data and HPC applications. These models can help *Eley* to choose the optimal operator for prefetching that improves the Big Data application performance as much as possible without violating the performance requirements of HPC applications.

Acknowledgment

The corresponding author is Shadi Ibrahim (shadi.ibrahim@inria.fr). This work is partially supported by the ANR KerStream project (ANR-16-CE25-0014-01). The experiments presented in this paper were carried out using the Grid'5000/ALADDIN-G5K experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see http://www.grid5000.fr/ for details).

REFERENCES

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] H. Jin, S. Ibrahim, L. Qi, H. Cao, S. Wu, and X. Shi, "The MapReduce Programming Model and Implementations," *Cloud Computing: Principles and Paradigms.*, pp. 373–390, 2011.

[3] "CMU OpenCloud Hadoop Cluster," http://ftp.pdl.cmu.edu/pub/datasets/hla/dataset.html, Accessed on Feb 2017.

[4] NICS, "Kraken Cray XT5 system, http://www.nics.tennessee.edu/computing-resources/kraken."

[5] NCSA, "BlueWaters project, http://www.ncsa.illinois.edu/BlueWaters/."

[6] "Big Data and Extreme-scale Computing (BDEC) Workshop, http://www.exascale.org/bdec/."

[7] G. Fox, J. Qiu, S. Jha, S. Ekanayake, and S. Kamburugamuve, "Big Data, simulations and HPC convergence," in *Workshop on Big Data Benchmarks*. Springer, 2015, pp. 3–17.

[8] Y. Guo, W. Bland, P. Balaji, and X. Zhou, "Fault tolerant MapReduce-MPI for HPC clusters," in *International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 34.

[9] S. Venkataraman, A. Panda, G. Ananthanarayanan, M. J. Franklin, and I. Stoica, "The power of choice in data-aware cluster scheduling," in *International Conference on Operating Systems Design and Implementation*. USENIX Association, 2014, pp. 301–316.

[10] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris, "Scarlett: coping with skewed content popularity in MapReduce clusters," in *European Conference on Computer Systems*. ACM, 2011, pp. 287–300.

[11] S. Ibrahim, H. Jin, L. Lu, B. He, G. Antoniu, and S. Wu, "Maestro: Replica-aware map scheduling for MapReduce," in *International Symposium on Cluster, Cloud and Grid Computing*. IEEE/ACM, 2012, pp. 435–442.

[12] O. Yildiz, M. Dorier, S. Ibrahim, R. Ross, and G. Antoniu, "On the root causes of cross-application I/O interference in HPC storage systems," in *International Parallel and Distributed Processing Symposium*. IEEE, 2016, pp. 750–759.

[13] N. S. Islam, M. Wasi-ur Rahman, X. Lu, and D. K. Panda, "High performance design for HDFS with byte-addressability of NVM and RDMA," in *International Conference on Supercomputing*. ACM, 2016, p. 8.

[14] N. Chaimov, A. Malony, S. Canon, C. Iancu, K. Z. Ibrahim, and J. Srinivasan, "Scaling Spark on HPC systems," in *International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 2016, pp. 97–110.

[15] K. Sato, K. Mohror, A. Moody, T. Gamblin, B. R. de Supinski, N. Maruyama, and S. Matsuoka, "A user-level infiniband-based file system and checkpoint strategy for burst buffers," in *International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2014, pp. 21–30.

[16] Y. Wang, R. Goldstone, W. Yu, and T. Wang, "Characterization and optimization of memory-resident MapReduce on HPC systems," in *International Parallel and Distributed Processing Symposium*. IEEE, 2014, pp. 799–808.

[17] "Hadoop Workload Analysis," http://www.pdl.cmu.edu/HLA/index.shtml, Accessed on Jan 2017.

[18] INRIA, "Grid'5000: http://www.grid5000.fr."

[19] R. B. Ross, R. Thakur *et al.*, "PVFS: A parallel file system for Linux clusters," in *Annual Linux Showcase and Conference*, 2000, pp. 391–430.

[20] "HiBench Big Data microbenchmark suite, https://github.com/intel-hadoop/HiBench."

[21] H. Shan and J. Shalf, "Using IOR to Analyze the I/O Performance for HPC Platforms," in *Cray User Group Conference*, Seattle, WA, USA, 2007.

[22] M. Dorier, G. Antoniu, F. Cappello, M. Snir, R. Sisneros, O. Yildiz, S. Ibrahim, T. Peterka, and L. Orf, "Damaris: Addressing performance variability in data management for post-petascale simulations," *ACM Transactions on Parallel Computing (TOPC)*, vol. 3, no. 3, p. 15, 2016.

[23] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng, "Datastager: scalable data staging services for petascale applications," in *International Symposium on High Performance Distributed Computing*. ACM, 2009, pp. 39–48.