Resource Distribution Estimation for Data-Intensive Workloads: Give Me My Share & No One Gets Hurt!

Alireza Khoshkbarforoushha, Rajiv Ranjan, and Peter Strazdins

Australian National University, Canberra, Australia
CSIRO, Canberra, Australia
{alireza.khoshkbarforoushha,raj.ranjan}@csiro.au,
peter.strazdins@cs.anu.edu.au

Abstract. Robust resource share estimation of data-intensive workloads is integral to efficient workload management in a cluster where multiple systems co-exist and share the same infrastructure. However, developing a reliable resource estimator is quite challenging due to (i) heterogeneity of workloads (e.g. stream processing, batch processing, transactional, etc.) in a multi-system shared cluster, (ii) limited (in batch processing) or complete uncertainties (in stream processing) on input data size or arrival rates, and (iii) changing configurations from run to run. To address above challenges, we propose an inclusive framework and related techniques for *workload profiling*, *similar job identification*, and *resource distribution prediction* in a cluster. Our analysis shows that the framework can successfully estimate the whole spectrum of resource usage as probability distribution functions for wide ranges of data-intensive workloads.

Keywords: Resource estimation, Big Data workload, Multi-Cluster workload management, Distribution prediction, Data-intensive systems.

1 Introduction

Datacenter-scale computing for big data analytics workload has seen a surge in adoption in the recent past due to availability and affordability of large-scale data processing systems which transform the traditional data mining and machine learning techniques into easy to program and deploy distributed analytics applications. Following the *no one size fits all* philosophy, currently many big data systems (e.g. Apache Hadoop, Apache Spark, Apache Mahout, etc.) are deployed in a cluster in which they share the same infrastructure under a set of sharing policies (e.g. fair and capacity scheduling supported by well-known Yarn [18] cluster resource management framework).

In a multi-system cluster with possibly hundreds of thousands of daily jobs, characterizing a typical workload is extremely challenging. However, in the literature they are classified as either productions or best effort jobs [5]. As the case in point, Hive workloads itself can be categorized as two classes of queries

i) HiveQL queries that are run repeatedly under the same system configuration parameters but on new datasets. Examples of such analytics include reporting queries, log analysis queries, Extract Transform Load (ETL) workloads, and ii) Interactive best-effort jobs which are typically submitted by data scientists to evaluate ideas.

Production jobs (e.g. Oozie), unlike the best-effort ad-hoc jobs, are business-critical, meaning that missing their service level agreements (SLAs) can have substantial financial impact. Moreover, this class of workloads typically consume more than 90% of the big data cluster resources [5]. State of the art workload scheduling techniques [5, 13, 8] focus on guaranteeing the SLAs for this class of workload subject to fairness, capacity, priority, and throughput maximization. Many of them [5, 8] need the cost a workload to be specified a priori to be able to define appropriate resource sharing policies.

However, developing a reliable resource estimator for *production jobs* is a hard research problem due to: (i) heterogeneity of workloads pertaining to each class of big data systems, (ii) limited (in batch processing) or complete uncertainties (in stream processing) on input data size/arrival rates and schema, and (iii) changing configurations (e.g. number of mappers/reducers or spouts/bolts respectively in Hadoop and Storm) from run to run. To address above problems, we propose an inclusive framework and related techniques for resource distribution estimation of heterogeneous big data workloads in a shared cluster.

To this end, the proposed framework first generates a set of data-intensive specific job templates (JT) by applying a clustering technique on a set of job characteristics. These templates are then used to identify similar jobs. Once the templates are generated, an statistical machine learning (ML) model is built for each of them by exploiting the past execution traces within the template. We argue that the existing single point resource estimator is not adequate for describing the whole spectrum of resource usage of big data workloads. Therefore, we introduce the novel approach of applying mixture density networks (MDN) as an underlying ML technique to approximate the probability distributions by means of finite mixture of Gaussians.

Therefore, the main contributions of the proposed resource estimation framework for big data analytics workload are: i) Introducing appropriate techniques to profile and identify similar jobs within a heterogeneous workloads from both batch and stream data processing systems, ii) Proposing a novel approach of estimating the full spectrum of resource usage in form of probability density functions (pdfs) as opposed to the single point conditional average.

The reminder of the paper is organized as follows: The next section explores the related work. Section 3 presents the overview of the proposed framework. In this section, we propose our ideas on *How to define templates for similar jobs identifications?* and *How to build resource distribution prediction models?*. Section 4 presents initial results on distribution based resource modelling along with some discussions on how the predicted pdfs can be utilized in workload management scenarios. The paper ends with some concluding remarks and the plan for future work in section 5.

2 Related Work

In this section we present a high level discussion on the past research done in the domain of job performance estimation. Following that, discussion of related work including performance estimation of declarative (SQL-style) and procedural (MapReduce style) workload is presented.

Job Performance Estimation

Job performance estimation, in general, using historical information of *similar* job has been studied in the past in domain of parallel computing [17, 16]. The key difference among the existing approaches is the way they tackle the problem of identifying similar jobs. In particular, authors in [17] focus on application characteristics such as user of jobs, jobs' submission time, jobs' arguments and on the number of physical servers on which the jobs are submitted for the definition of application similarity. Recently, authors in [16] propose the novel use of clone detection technique to determine the clone level of a newly submitted job with respect to the jobs in the execution history and to predict the resource requirement of the new job depending on its clone level.

High-level Data Intensive Frameworks

There are a number of related work on runtime and resource usage estimation in the context of DBMS [1, 12]. In the majority of related work, different statistical ML techniques are applied for estimating query performance. These approaches typically build statistical models using past query executions and a representative set of query features (query plan and/or operator level features) which possibly have the high predictive power in terms of resource or performance measures.

When it comes to MapReduce ecosystem, a major fraction of big data cluster workloads are generated by a handful of high-level frameworks such as Hive, Pig, Giraph [4]. This opens up an opportunity to train ML techniques against a set of finite recurring operators to estimate workload performance. For example, authors in [7] apply the Kernel Canonical Correlation Analysis (KCCA) to the Hive execution plan operators. They conclude that *only* a set of *low level features* pertaining to Hive query execution such as the number of maps and reduces, bytes read locally, bytes read from HDFS lead to accurate performance modelling. However, the provided low level features are *not* available before actual query execution. Therefore, above technique cannot be applied for performance prediction of new incoming workloads.

Similarly, authors in [15] propose a technique that predicts the runtime performance for a fixed set of queries running over varying input data sets. Specifically, it splits each query into several segments where each segment's performance is estimated using uni-variate linear regression. Next the estimates are plugged into a global analytical model to predict the overall query runtime. Since modelling a small and finite space of relational operators might not be adequate for all MapReduce workloads (e.g. iterative analytics), several studies focused on more fine-grained analysis of MapReduce job performance analysis as discussed

next.

Procedural Data Intensive Workload

Herodotou et al. [9] propose a self-tunning system for Hadoop that uses performance models with the goal of workload tuning, finding the best configuration settings for a given workload and a cluster infrastructure. Along these lines, the authors in [19] first build the job profile from the job past executions or by executing the workload on a smaller data set using an automated profiling tool. Then, they apply the performance bounds of completion time of different job phases to predict the job completion time as a function of the input dataset size and allocated resources.

In terms of runtime prediction of ML algorithms executing on top of MapReduce ecosystem, authors in [14] present an experimental methodology for predicting the runtime of iterative algorithms written in Apache Giraph. To do so, they conduct sample runs for capturing the algorithm's convergence trend and per-iteration key input features.

Concluding Remarks

In summary, all of the above studies focused on single type of workload in particular Hadoop, while in reality big data workloads are heterogeneous consisting of multiple types of systems (e.g. Apache Hive, Apache Storm, etc.) and jobs. In contrast, our framework considers heterogeneous workloads where different jobs and queries from either batch or stream data processing systems running side by side. Moreover, existing approaches estimate resource and performance as a single point value which is neither expressive enough nor does it capture the possible variances due to resource contentions and interferences from other workloads. In contrast, we use a distribution prediction technique that describes the resource usage as conditional distribution functions.

3 Overview of the Proposed Framework

The problem of resource requirement estimation for future job in a multi-system cluster is decomposed in i) characterizing the similar jobs that have executed in the past, and ii) building a prediction model based on the collected statistics. The workflow in the proposed framework, as shown in Figure 1, is as follows. Firstly, workload profiling is conducted in order to collect required information for defining similar jobs and building resource models. Secondly, a set of job templates are generated based on the collected features and their corresponding values. Finally, a distribution ML model is trained and built for each job templates which is responsible for resource prediction of a new incoming workload.

3.1 Similarity Definition and Template Generation

The routine nature of productions jobs allows us to build the resource model based on the past execution profiles of the *same* or *similar* jobs. Thus, building

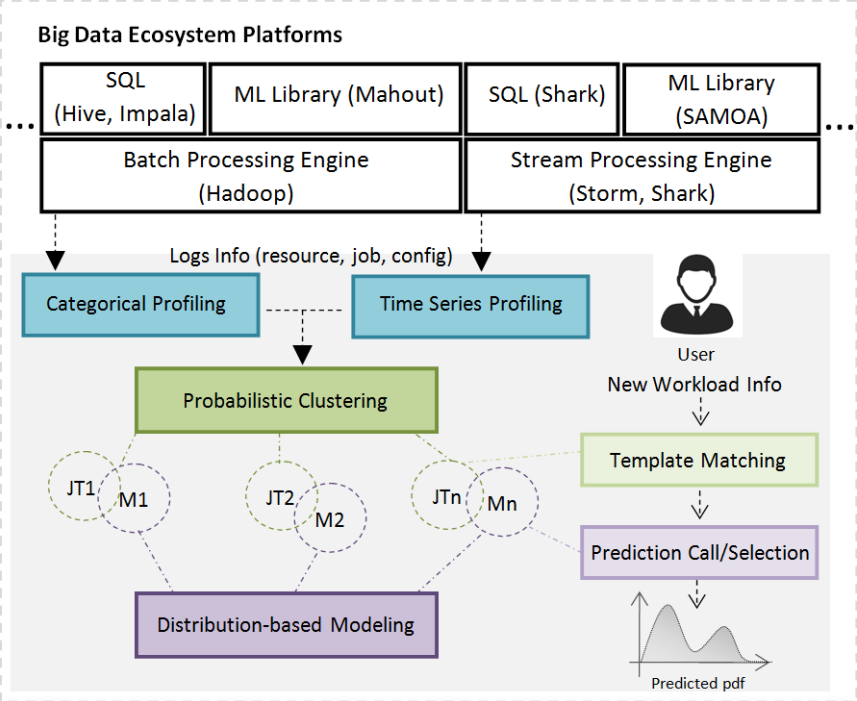

Fig. 1: Comprehensive framework for performance distribution prediction of heterogeneous data-intensive workloads.

profiles is the first step. Our framework considers both batch and streaming workloads, thereby the profiles will be logged as either categorical or time series files respectively. We next look into *what* to profile.

Resource usage r is the function of a job j that is executed on input data (stream) d using configuration parameter settings c:

$$r = f(j, d, c) \tag{1}$$

This means that resource modelling requires logging appropriate information about job, its data flow and configurations along with the resources in use, as follows:

– **Job Profile**
  - *Input Data Stats:* such as number of Input Records/Bytes (e.g. in Hive tables), Average arrival rates (in Stream processing), Data file format, etc.
  - *Job Metadata:* including Job Name, User who submitted the job, Submission time, File input path, and so on.
  - *Runtime Stats:* such as number of mappers and reducers per stage (as in Hive), Response time, Latency/Throughput (in stream processing).

- **Resource Profile** including CPU time and utilization, Memory usage, local/network I/O.
- **Configuration Profile**
  - *Compact*[1] *job configs:* such as number of reducers in Hive which can be set for a specific query using mapred.reduce.tasks property and override the cluster wide settings.
  - *Compact cluster configs:* such as the number of virtual CPU cores for each reduce task of a job which is set by mapreduce.reduce.cpu.vcores property in Apache Yarn [18].

The job profiling is a recurrent process, means that the mentioned profiles will be generated for every submitted job. However, to avoid any performance degradation, two design principles need to be realized. First, statistics should be collected passively, without affecting the performance. For example, in [11] we used *dstat* (http://dag.wiee.rs/home-made/dstat/) as a lightweight python-based tool that collects OS and system statistics non-intrusively. Second, profiling process should be tractable via enabling a feature to on/off profiling process. This means that every job is profiled unless it is deactivated by the user of the final prototype.

Upon extracting required information from job history and big data cluster logs, we now focus on the second challenge, that is characterizing the similar jobs that have executed in the past. We need to identify a set of classes of jobs (i.e. job templates) that exhibit a similar resource usage pattern.

Modern big data clusters run a diverse mix of applications and production workloads [18], thereby characterizing similar jobs is challenging. Although difficult, we argue that appropriate clustering techniques along with the proper job execution and big data system configuration profiles lead to formation of fitting templates.

In our problem domain the clusters (i.e. candidate job templates) need not be disjoint, and the same job can be associated to several classes. Because two jobs can be compared in many ways. For example, our initial analysis on synthetic MapReduce workloads of Facebook [2] [4] demonstrates that the (submit_time_seconds, hdfs_input_path) is a proper candidate job template since jobs with the close submission time and same input path have roughly same map input byte size, shuffle bytes, etc. Yet another template is (user, submit_time_seconds, hdfs_input_path) which is more restricted. Intuitively, a certain job can be assigned to both of these templates with respect to its characteristics.

Therefore, we use a probabilistic clustering technique, Expectation Maximization (EM) algorithm [6] which is a soft clustering technique. EM finds clusters by determining a mixture of Gaussians that fit a given data set. Each Gaussian has a mean and covariance matrix. The prior probability for each Gaussian

[1] Due to the large number of configuration parameters, only a subset of settings which have substantial impacts on resource and performance measures need to be logged.
[2] https://github.com/SWIMProjectUCB/SWIM/wiki

is the fraction of points in the cluster defined by that Gaussian. These parameters can be either initialized by randomly selecting means of the Gaussians or by using the output of K-means algorithms for the initial centres. EM converges on a locally optimal solution by iteratively updating means and variances.

Once the candidate job templates (i.e. clusters) becomes available, different validity measures which falls broadly into internal, relative, and external validations are used to evaluate clustering results. The output of this step is a set of final job templates through which we are able to identify who belongs to whom. To do so, for a new job, the template membership probability is calculated. The higher probability seemingly shows the more the templates match the job's characteristics.

3.2 Distribution Prediction

Many workload scheduling studies [8, 5] formulate and optimize the policies subject to fairness, capacity, and priority, conditioned on having the cost of a workload a priori in a multidimensional space representing different resources. In contrast, there exists criticism of the off-line predictions [13] due to the interferences from other workloads concurrently running and sharing the same resources at runtime.

As stressed in related work, the state of the art resource and performance estimation techniques for data intensive workloads only provide the conditional mean of the point of interest. However, even running the same query on the same data with constant configuration show different performance and resource behaviour. In response, we argue that the *distribution* estimation provides an expressive description of the target values (e.g. runtime, CPU time) and the possible variances due to resource contention.

We adopt a novel approach of workload resource distribution prediction using Mixture Density Networks (MDN) [3]. An MDN fuses a Gaussian mixture model (GMM) with feed-forward neural networks. In MDN, the distribution of the outputs t is described by a parametric model whose parameters are determined by the output of a neural network, which takes x as inputs. Specifically, an MDN maps a set of input features x to the parameters of a GMM including mixture weights α_i, mean μ_i, and variance σ^2 which in turn produces the full *pdf* of an output feature t, conditioned on the input vector. Detailed discussion on the proposed approach are available in accompanying technical reports [10, 11] which discuss how to predict the resource and performance distribution for batch (Hive workloads) [10] and stream data processing (i.e. continuous queries) [11]. We will show the efficacy of distribution as opposed to existing techniques in workload resource modelling in the next section.

Note that due to the large number of profiles and historical logs, building a model could be prohibitively expensive, though we already showed [10] that the training time of the MDN linearly grows with respect to the training data size. Thus, enabling a *maximum history* feature as in [17] which indicates the maximum number of data points to be used for building/refreshing a model is inevitable.

Once the model is built and trained, it can then be invoked when a similarity measures assign a new job to one of the existing classes. Since the job templates overlap, a single job may be associated to multiple job templates and their corresponding ML model. Therefore, the prediction with the smallest confidence interval will be selected.

4 Initial Results

In this section, we present some compact yet lucid results on how the distribution prediction look like and how one can utilize them for appropriate policy setting across a shared big data cluster.

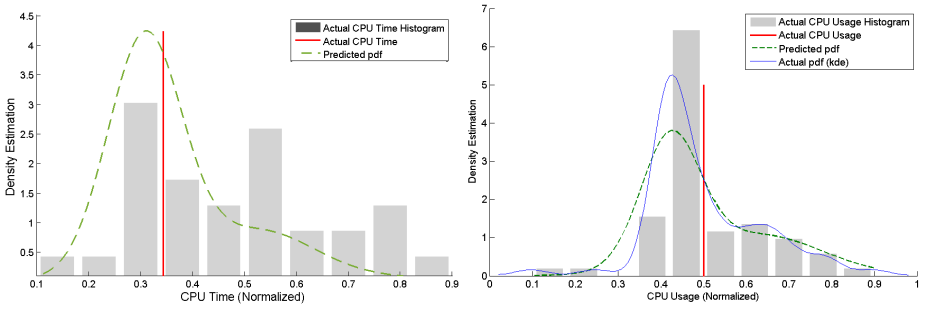

Fig. 2: (a) A sample predicted pdf for CPU Time of an input test from Hive workload. (b) A sample predicted pdf for CPU Utilization, selected from a test dataset of linear road benchmark.

Figure 2(a) plots a sample predicted pdf for CPU time for one of the experiments conducted on TPC-H in [10]. The predicted pdf is corresponding to a test input from Template-7 (Q7) of TPC-H against 100GB database size. To demonstrate the whole possible range of CPU time values under Q7, the histograms for 30 instance queries based on Q7 from test set are shown as well.

As we can see, the predicted distribution adequately estimates the CPU time distribution in which they show high probability around the target value. More importantly, they provide information about the whole spectrum of resource usage. In particular, the predicted pdf shows highly probable CPU time in ranges (0.25, 0.4) which are consistent with the actual distribution. Note that the predicted pdf is concerned with one single input, thereby the resulted uncertainty of pdf for the range (0.6, 0.9) is justifiable.

In a similar manner, Figure 2(b) depicts a sample predicted pdf and actual CPU usage in terms of normalized histogram and fitted kernel density estimation (kde) for one of the experiments on linear road benchmark [2] queries conducted in [11]. As the figure indicates, the estimated pdf approximates the actual resource usage pdf closely. The predicted pdf provides a complete description of

the statistical properties of the CPU utilization through which we are not only able to capture the observation point, but also the whole spectrum of the resource usage. In contrast, a best estimation from the existing resource estimation techniques [1, 7, 9, 12, 19] merely provides the point which is visualized by solid vertical line. Unlike pdfs, with such estimation we are not able to directly calculate any valuable statistical measures (e.g. variance, expectation, confidence interval) about the target data. Detailed evaluation of the proposed approach and its comparison with the state of the art single point estimator can be found in accompanying technical reports [10, 11].

Once the resource usage distribution becomes available, it can then be used to define appropriate resource pool policy for critical SLA-driven workloads. For example, Cloudera Manager [3] provides the ability to statically allocate resources using Linux control groups, through which one can allocate services (e.g. Hive, Storm, Spark) a *percentage* of total resources. Static resource pools isolate the services in the cluster from one another, so that load on one service has a bounded impact on the others. With distribution based resource estimation, we are able to determine appropriate percentage of resource shares for a workload before actual execution, without sacrificing the cluster throughputs and utilization.

5 Conclusions and Future Work

This paper proposed an inclusive framework and related techniques for resource usage distribution prediction of heterogeneous big data workloads in a cluster. To this end, our framework uses the clustering techniques along with the statistical machine learning algorithm (i.e. MDN) to identify similar jobs and build a distribution-based prediction models. The initial results show that the approach is capable of estimating resource usage distribution accurately, through which we are able to define more reliable resource sharing policies aiming at guaranteeing SLA subject to fairness, capacity, priority, and throughput maximization.

As an ongoing work, we plan to complete and evaluate the template generation phase using real-world workload traces collected from our private CSIRO big data cluster and possibly more traces from other companies. Following that, we also plan to accommodate the distribution predictions in cost-optimized resource provisioning of big data analytics flows in a datacenter cloud.

References

1. M. Akdere, U. Çetintemel, M. Riondato, E. Upfal, and S. B. Zdonik. Learning-based query performance modeling and prediction. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 390–401. IEEE, 2012.
2. A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear road: a stream data management benchmark. In

[3] http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise/cloudera-manager.html

Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, pages 480–491. VLDB Endowment, 2004.
3. C. M. Bishop. Mixture density networks. 1994.
4. Y. Chen, S. Alspaugh, and R. Katz. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *VLDB*, 5(12):1802–1813, 2012.
5. C. Curino, D. E. Difallah, C. Douglas, S. Krishnan, R. Ramakrishnan, and S. Rao. Reservation-based scheduling: If you're late don't blame us! In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–14. ACM, 2014.
6. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.
7. A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson. Statistics-driven workload modeling for the cloud. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, pages 87–92. IEEE, 2010.
8. A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *NSDI*, volume 11, pages 24–24, 2011.
9. H. Herodotou and S. Babu. Profiling, what-if analysis, and cost-based optimization of mapreduce programs. *VLDB*, 4(11):1111–1122, 2011.
10. A. Khoshkbarforoushha and R. Ranjan. Resource and performance distribution prediction for large scale analytics (hive) queries. *TR-2015-01, https://rranjans.files.wordpress.com/2015/05/tpctc.pdf*, 2015.
11. A. Khoshkbarforoushha, R. Ranjan, R. Gaire, P. P. Jayaraman, J. Hosking, and E. Abbasnejad. Resource usage estimation of data stream processing workloads in datacenter clouds. *arXiv preprint arXiv:1501.07020*, 2015.
12. J. Li, A. C. König, V. Narasayya, and S. Chaudhuri. Robust estimation of resource consumption for sql queries using statistical techniques. *Proceedings of the VLDB Endowment*, 5(11):1555–1566, 2012.
13. J. Mace, P. Bodik, R. Fonseca, and M. Musuvathi. Retro: Targeted resource management in multi-tenant distributed systems. In *NSDI*. USENIX, 2015.
14. A. D. Popescu, A. Balmin, V. Ercegovac, and A. Ailamaki. Predict: towards predicting the runtime of large scale iterative analytics. *Proceedings of the VLDB Endowment*, 6(14):1678–1689, 2013.
15. A. D. Popescu, V. Ercegovac, A. Balmin, M. Branco, and A. Ailamaki. Same queries, different data: Can we predict runtime performance? In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*, pages 275–280. IEEE, 2012.
16. M. Sarkar, T. Mondal, S. Roy, and N. Mukherjee. Resource requirement prediction using clone detection technique. *Future Generation Computer Systems*, 29(4):936–952, 2013.
17. W. Smith, I. Foster, and V. Taylor. Predicting application run times using historical information. In *Job Scheduling Strategies for Parallel Processing*, pages 122–142. Springer, 1998.
18. V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.
19. A. Verma, L. Cherkasova, and R. H. Campbell. Aria: automatic resource inference and allocation for mapreduce environments. In *Proceedings of the 8th ACM international conference on Autonomic computing*, pages 235–244. ACM, 2011.