An Online Data Access Prediction and Optimization Approach for Distributed Systems

Renato Porfírio Ishii and Rodrigo Fernandes de Mello

Abstract—Current scientific applications have been producing large amounts of data. The processing, handling and analysis of such data require large-scale computing infrastructures such as clusters and grids. In this area, studies aim at improving the performance of data-intensive applications by optimizing data accesses. In order to achieve this goal, distributed storage systems have been considering techniques of data replication, migration, distribution, and access parallelism. However, the main drawback of those studies is that they do not take into account application behavior to perform data access optimization. This limitation motivated this paper which applies strategies to support the online prediction of application behavior in order to optimize data access operations on distributed systems, without requiring any information on past executions. In order to accomplish such a goal, this approach organizes application behaviors as time series and, then, analyzes and classifies those series according to their properties. By knowing properties, the approach selects modeling techniques to represent series and perform predictions, which are, later on, used to optimize data access operations. This new approach was implemented and evaluated using the OptorSim simulator, sponsored by the LHC-CERN project and widely employed by the scientific community. Experiments confirm this new approach reduces application execution time in about 50 percent, specially when handling large amounts of data.

Index Terms—Distributed computing, distributed file system, data access optimization, time series analysis, prediction.

1 INTRODUCTION

Scientific applications tend to produce and handle large amounts of data. Examples of such applications include the Pan-STARRS project,1 which captures at about 2.5 petabytes (PB) of data per year, and Large Hadron Collider project (LHC),2 which generates from 50 to 100 PB of data every year. Those applications tend to consider distributed computing tools to deal with the need for high performance and storage requirements. Such tools have great potential for solving a vast class of complex problems; however, they are still limited in terms of manipulating large amounts of data [1].

Those applications rely on cluster and grid environments, which are the main large-scale computing infrastructures currently available. Clusters are composed of a set of workstations or personal computers with similar architecture, connected via a high-speed local area network, in which every station usually executes the same operating system. Grids are organized as federations of computers, in which one federation is a different administrative domain sharing its own resources. This type of infrastructure is characterized by hardware and software heterogeneity [2].

These infrastructures meet the demands imposed by several types of applications, however, there is still room for significative theoretical and practical improvements when dealing with large amounts of data. Those improvements would certainly motivate the adoption of distributed computing in several domains such as high energy physics, financial modeling, earthquake simulations, and climate modeling [3], [4], [5], [6]. However, there are still great challenges to meet such improvements, which comprise dealing with heterogeneous resources, access latency variations, fault detection, and recovery, etc.

All those challenges have been motivating studies in different subjects such as job scheduling, load balancing, communication protocols, high-performance hardware architectures, and data access optimization [4], [7]. This later subject, i.e., data access optimization, is particularly appealing to the Computer Science area, mainly due to the possibility to provide high performance to data-intensive applications [5], [6], [8].

This possibility has motivated the study of the Data Access Problem (DAP) in distributed computing environments [9]. The complexity of this NP-complete problem is justified by the optimization of data location and consistency, which involve operations such as data distribution, migration and replication [10]. Data distribution aims at efficiently allocating file chunks on the computing environment attempting to improve the performance of read-and-write operations. Migration is also considered in order to


R.P. Ishii is with the Faculty of Computer Science, Federal University of Mato Grosso do Sul-UFMS, Cidade Universitária, Caixa Postal 549, Campo Grande, MS 79070-900, Brazil. E-mail: renato@facom.ufms.br.

R.F. de Mello is with the Department of Computer Science, Institute of Mathematics and Computer Sciences, University of São Paulo-USP, Avenida Trabalhador sãocarlense, 400, Centro, Caixa Postal 668, São Carlos, SP 13560-970, Brazil. E-mail: mello@icmc.usp.br.

Manuscript received 17 May 2010; revised 19 Aug., 2011; accepted 23 Sept., 2011; published online 19 Oct., 2011. Recommended for acceptance by I. Stoimenovici.

For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2010-05-0299.

Digital Object Identifier no. 10.1109/TPDS.2011.256.
bring data closer to where it is requested. In that sense, it must analyze data transfer impacts to reduce costs of access operations. Replication takes decisions about where and when to copy files (create replicas) as a way to increase performance and resilience against failures [11]. Complementarily, consistency aims at coordinating the update of multiple file replicas [12], providing a single data view to distributed applications.

All those factors have been studied and possible solutions proposed, as presented in Section 2, however they still lack in three substantial points: 1) most works only consider read operations, what tends to limit their applicability in real-world scenarios; 2) they employ static approaches to optimize access, i.e., mechanisms that do not consider the dynamic system behavior; finally, 3) works still need to employ an extra effort in reducing application execution time, what depends on data access patterns. In this context, it is very important to understand, analyze, estimate, and predict application behavior in order to optimize read-and-write operations, consequently, reducing execution time. The term behavior in here characterizes data access operations over time, including, for example, the operation type (read or write), which and when a file is accessed, etc.

Given such drawbacks, this paper employs strategies to support the online prediction of application behavior in attempt to optimize data access operations on distributed systems. In order to accomplish this goal, our approach considers the following steps

1. Application knowledge acquisition;
2. Organization of process behaviors as time series;
3. Analysis of time series generation processes;\(^3\)
4. Selection of techniques to model times series;
5. Definition of how many future observations will be predicted;
6. Prediction of observations, and, finally,
7. Execution of the optimization heuristic in attempt to reduce the time consumed in data access operations.

We designed this approach in order to automatize all those steps, avoiding or at least expressively reducing human intervention.

Experiments were conducted using the OptorSim simulator [13], sponsored by the LHC/CERN project and widely employed by the scientific community [9], [14], [15], [16], [17], [18], [19]. Results confirm this new approach reduces application execution time in about 50 percent, specially when dealing with large amounts of data.

The remainder of this paper is organized as follows: related works are presented in Section 2; Section 3 presents the time series classification approach and experiments on real data to confirm it is capable of selecting adequate modeling techniques [20]; the new optimization heuristic is presented in Section 4 and experimental results are shown in Section 5; Finally, we present concluding remarks and references.

3. In Time Series Analysis, a system is assumed to produce a particular time series. In that context, term generation process is related to such system properties, i.e., stochasticity, linearity and stationarity.

2 RELATED WORKS

This section presents related works on the prediction of application behavior and data access optimization approaches.

2.1 Prediction of Application Behavior

Several studies have been considering statistical techniques to analyze data and construct probabilistic models to characterize and predict application workloads in distributed environments. Those models have been employed to assist fault diagnosis, resource allocation, and system performance optimizations.

As one of the first works in this area, Devarakonda and Iyer [21] propose a statistical approach to predict the consumption of CPU, file system I/O, and memory. This study models the behaviors of processes using automata, storing those models in databases. When a new process arrives at the system, the approach verifies if there is any automaton capable of representing it. If so, this automaton is used to estimate resource requirements for this next process. Trace-driven experiments confirm a strong correlation in between next and past executions.

Faerman et al. [22] propose the Adaptive Regression Modeling (AdRM) to estimate file transferring events of distributed applications. AdRM was evaluated considering two data-intensive applications: the Synthetic Aperture Radar Atlas (SARA) and the Storage Resource Broker (SRB). SARA handles a large number of small files (from 1 to 3 MB), while SRB deals with files at about 16 MB. Experiments compare AdRM against the Raw Bandwidth model (RBW). Results confirm AdRM presents a good prediction accuracy, however, this work lacks in comparisons and analysis to other well-known models.

Wang et al. [23] employ machine learning techniques to model requests to storage devices, based on Classification and Regression Trees (CART). This work proposes two prediction approaches: 1) the first considers one prediction model for every data access request; 2) the second models and predicts observations based on an average behavior of requests. Experiments show that both models present an average error of 17 and 38 percent, respectively. The first is computationally costly though.

Senger et al. [24] propose an online approach to acquire, classify, and extract process behavior. The information acquisition is obtained by instrumenting the Linux kernel. In this sense, one does not need to modify nor recompile user applications in order to monitor them. This information is submitted to an ART-2A neural network architecture, which groups and labels process behavioral states. Results confirm the approach is capable of automatically modeling process behaviors, also presenting good computational performance, stability, and plasticity.

Senger et al. [25] also propose an approach to predict execution times of parallel applications, aiming at improving scheduling decisions in large-scale environments. This approach provides application knowledge to the system scheduler, which is responsible for optimizing resource allocation. According to experiments, this approach predicts the average behavior of applications presenting errors in range [38, 57] percent.
ISHII AND DE MELLO: AN ONLINE DATA ACCESS PREDICTION AND OPTIMIZATION APPROACH FOR DISTRIBUTED SYSTEMS

Mello [26] employ Chaos-Theory concepts and nonlinear prediction techniques to model and predict process behavior over time. Results confirm behaviors can be modeled using Dynamical Systems and, particularly, that Radial Basis Functions are good estimators for future process states. Schedulers can take advantage of that study to improve resource allocation.

The studies mentioned in this section demonstrated performance improvements in many scenarios, such as resource allocation, fault diagnosis, and others. The main goal of such works is to apply different approaches to analyze data and construct probabilistic models to characterize and predict application workloads in distributed environments. One of the main drawbacks of those studies is that they do not evaluate properties related to application behaviors (such as stochasticity, linearity, and stationarity), therefore, they do not provide or select adequate tools to model and predict such behaviors. Moreover, they do not consider an automatic and complete approach from behavior extraction to optimization. Such drawbacks are here addressed as detailed in next sections.

2.2 Data Access Optimization

Several studies have been attempting to improve data access on distributed environments. Such works are mainly focused on data replication, distribution, and consistency. In this context, Oldfield and Kotz [27] proposed the Armada framework to execute, control, and monitor applications. Armada builds graph structures to represent processing and data flows. Graphs, representing process versus data dependencies, support decisions on moving data toward processes, thus reducing execution time. Experiments compare applications running on a traditional environment and also on Armada. In that scenario, Armada improves network throughput in around 40 percent.

Kim et al. [28] propose a heuristic to improve data access, i.e., reduce execution time of data-intensive applications. This heuristic attempts to find the lowest cost data source for every process. Authors consider a trace-based synthetic scenario on PlanetLab to evaluate their approach. Results confirm that heuristic outperforms conventional latency-based and random allocation approaches.

Chervenak et al. [29] propose a framework called Replica Location Service (RLS) which maintains and provides information on physical locations of replicas. RLS is used in a variety of production environments such as the Laser Interferometer Gravitational Wave Observatory (LIGO) [30], Earth System Grid (ESG) [31], and Pegasus [32]. Authors presented a performance study demonstrating that the individual RLS servers perform and scale up well when compared to the native MySQL using ODBC clients.

AL-Mistarihi and Yong [14] propose an approach to select the best replica, i.e., which presents the lowest access cost, for applications. Authors consider the Analytical Hierarchy Process (AHP) to solve that problem, they evaluate this approach using the OptorSim simulator and compare it to a random one. Unfortunately, they did not compare the approach to the most common ones already included in OptorSim: LRU, LFU, and the Economic Model (ECO).

The main drawback of those works is that they do not take into account the dynamic behavior of applications, predictions nor estimations to perform data access optimization. These issues are considered in this paper.

3 TIME SERIES ANALYSIS

By modeling the outputs produced by real-world systems, we can study and, therefore, understand how they work and behave under different circumstances. This is specially interesting to support the prediction of behavior and, consequently, support decision making, what is particularly required in certain application domains. Here, we are interested in predicting read-and-write operations in attempt to optimize data access on distributed environments.

Outputs produced by real-world systems present a strong temporal dependency, i.e., adjacent observations are dependent [33]. This dependency strongly reduces the modeling accuracy of conventional techniques. In order to overcome this problem, a new area was developed, called Time Series Analysis [34], in which data are commonly organized in terms of variables and their observations over time.

Before modeling, nevertheless, it is necessary to understand the implicit features embedded in data such as the stationarity, linearity, and stochasticity. By understanding those features, we can better select modeling techniques to perform prediction. A time series is said to be stationary when its observations are in a particular state of statistical equilibrium, i.e., they evolve over time around a constant average and variance [34]. On the other side, in linear time series, observations are composed of a linear combination of past occurrences and noises [35]. Finally, stochastic time series are formed by random observations and relations, which follow probability density functions and may change over time [34].

The identification of these features, however, is not a simple task and requires not only specialists’ opinions but also additional information on how series were obtained, what supports the characterization of the time series generation process, i.e., the system used to produce such series. By knowing such generation process, we can select an adequate technique to model and study time series. However, the main problem of this approach is the subjectivity imposed by specialists, which can lead to failures. Moreover, during time series modeling, mainly when considering those obtained from real-world systems, researchers neither have further information nor specialists available to support this task [33].

These issues have been constantly limiting the appropriate modeling of series, mainly when researchers are not specialists in time series analysis or belong to other areas not related to statistics or mathematics, such as pharmaceutical research, biotechnology, medicine, public health, agriculture, and climate modeling. This is also the situation observed when studying process behavior, as in this paper. Such behavior can be organized as time series and, by understanding those series, we can better model and, consequently, predict observations.

The need for specialists and the lack of a step-by-step analysis on how to analyze time series motivated Ishii et al. [20] to propose an automatic and systematic approach to
evaluate and classify time series generation processes. This approach combines dynamical systems, statistical tools, and tests to support time series classification. All tools are organized according to a tree view (Fig. 1) and also indicate models for every branch.

At the first tree level (Fig. 1), there is the time series. The second level separates deterministic from stochastic series. When a series is deterministic, it is better studied using Chaos-Theory tools, making unnecessary further evaluations. Still in the second level, we have the stochastic type of series. When under that branch, series still must be evaluated in order to select the best modeling approach. Thus, we proceed with the third level which verifies series linearity. Nonlinear series can be addressed by using nonlinear modeling approaches [36], [37], [38]. When linear, we still need to evaluate whether the series is stationary or not. When the series is stochastic, linear, and stationary, it can be modeled by using techniques such as Autoregressive models (AR ), Moving averages (MA), or Autoregressive moving average models (ARMA) [34]. When the series is stochastic, linear, and nonstationary, it can be modeled by using Autoregressive integrated moving average models (ARIMA) [34].

By using this tree (Fig. 1), Ishii et al. [20] suggest adequate modeling techniques for every branch, which are capable of dealing with the time series generation process.

In this paper, process behavior is represented by the time series having information on read-and-write operations. The generation processes of those time series are then analyzed as before mentioned and, thus, modeling techniques are selected according to the approach proposed by Ishii et al. [20]. Such techniques are employed to predict process behaviors over time.

4 ONLINE DATA ACCESS PREDICTION AND OPTIMIZATION APPROACH FOR DISTRIBUTED SYSTEMS

This paper proposes an approach to support the online prediction of application behavior in an attempt to optimize data access operations on distributed systems. At a first step, this approach employs time series analysis tools, detailed in Section 3, to evaluate process behavior and select modeling techniques, which are used to predict application behaviors. At last, predictions are used as input to a new data access optimization heuristic.

This approach is composed of the following steps (Fig. 2)

1. Application knowledge acquisition;
2. Organization of process behaviors as time series;
3. Analysis of time series generation processes;
4. Selection of techniques to model times series;
5. Definition of how many future observations will be estimated;
6. Prediction of observations; and, finally,
7. Execution of the optimization heuristic in attempt to reduce the time consumed in data access operations.

The first step, i.e., application knowledge acquisition, is responsible for monitoring process behavior by using event interception. The interception mechanism is associated with the process under execution. At every function call, this mechanism is informed, being capable of taking further decisions or simply log information. Different libraries and tools provide interception, the most common ones are the Unix DLSym [39] and Unix Ptrace [40].

DLSym intercepts only calls to dynamic functions, i.e., functions made available by shared libraries. When a program calls a function, DLSym intercepts the call and injects any code instead. In our approach, we simply inject a code to log all information about the call, i.e., function name, arguments, and return value. By using DLSym we can inject any code even in precompiled programs, thus, there is no need to modify and compile the application source code. Consequently, any procedure in a dynamic shared library can be intercepted, what helps to build monitoring tools. The second mechanism is Ptrace which transparently intercepts process signals and system calls. It is usually employed to build diagnosis and debug tools. Differently from DLSym, Ptrace can intercept any system call or signals from applications.

This paper considered the DLSym mechanism because it efficiently extracts information of data access such as operation type, file identifier, etc., (see Table 1). Moreover, DLSym has lower impacts on capturing information than Ptrace, as depicted in [9].

After extracting the application behavior, we transform the sequence of read-and-write events in a multidimensional time series. Every event is described by quintuple \( tr = \{ \text{pid}, \text{inode}, \text{amt}, \text{time}, \text{op} \} \) in which \( \text{pid} \) is the process identifier that performed the operation, \( \text{inode} \) is the file identifier, \( \text{amt} \) is the amount of data written or read, \( \text{time} \) is the system time when the operation occurred, and \( \text{op} \) is the operation performed (read or write).

Fig. 1. Classification of time series generation processes, adapted from [20].

Fig. 2. The main steps of online predictive approach.
A series containing \( \text{TR} = \{tr_0, tr_1, \ldots, tr_{u-1}\} \). After this organization, the multi-dimensional time series (see example in Table 1) is stored in a trace file.

The third step evaluates the generation process of time series \( \text{TR} \) according to specific properties: stochasticity, linearity, and stationarity (see Section 3). Having this evaluation, we are able to select an adequate tool to model series and, afterwards, predict observations.

As presented in Section 3, Ishii et al. [20] discuss and propose an automatic and systematic approach to evaluate and classify time series generation processes. In that study, authors confirm: 1) Recurrence Plots (RP) [41] are useful to verify stochastic levels of time series; 2) the White Neural Network (WNN) test [42] is useful to evaluate the time series linearity; and 3) Space-Time Separation Plots (STP) [43] evaluates series stationarity. That approach and tools are here considered to evaluate the generation process of time series.

Based on the evaluation of the time series generation process, we select an adequate modeling technique as detailed in Section 3. According to Ishii et al. [20], when the series is deterministic, a reconstruction is conducted by considering the Takens’ immersion theorem [44], which relates series observations over time. On the other hand, when the series is stochastic and nonlinear, the suggested techniques are radial basis function approximation, artificial neural networks, Autoregressive Conditional Heteroskedasticity (ARCH) [35], filters, and polynomial approaches [38]. When stochastic, linear, and stationary, the series can be modeled by using AR, MA, or ARMA [34]. Otherwise, when the series is stochastic, linear, and nonstationary, it is better modeled using ARIMA [34].

In the next step, we consider the adaptive sliding-window (ASW) mechanism proposed in [9] to estimate the number of time series observations to be predicted, based on process behavior changes. As observed in [9], if we do not consider such estimator and, thus, predict less observations, the optimization heuristic will be more frequently executed. On the other hand, if we predict more observations, process behavior may change and the exceeding predictions become inaccurate enough, risking optimization decisions. ASW considers the type, number of operations, and \( \beta \) parameter to adapt the window length according to consecutive homogeneous operations (i.e., reading or writing). Equation (1) defines the ASW, where \( W_{t+1} \) is the next window length, \( W_t \) is the current window length, \( Op \) is the number of homogeneous operations, and \( \beta \) is the factor which determines modifications in the window length.

\[
W_{t+1} = W_t \times (1 - \beta) + Op^{2.5} \times \beta.
\] (1)

For example, given \( W_1 = 10 \), \( Op = 4 \), and \( \beta = 0.10 \), we would obtain a next window length equals to \( W_2 = 9 \). Note that the window adapts according to the number of operations under the same type, see Fig. 3. On the other hand, when \( Op \) is equal to \( W_t \) (see \( W_4 \) in Fig. 3) the next window length increases considerably, \( W_5 = 11 \). Parameter \( \beta \) is experimentally defined (as presented in [9]).

Fig. 3 presents a sample trace file with process behavior information (we focused only in reading (r) and writing (w) operations). The first part of the Fig. 3 shows the initial window length (\( W_1 = 10 \)) and all consecutive operations. The second part shows how to compute of the next sliding window length based on (1). Finally, the last part shows the subsequent windows \( W_2 = 9 \), \( W_3 = 8 \), \( W_4 = 7 \), and \( W_5 = 11 \), which clearly demonstrates the adaptation of window length according to the behavior of reading and writing operations.

After the previous steps, the prediction is performed on the time series, which represents process behaviors. The measurement considered to evaluate prediction results was the Normalized Root Mean Squared Error (NRMSE), defined in (2), in which \( x_{\text{max}} \) is the maximum and \( x_{\text{min}} \) is the minimum observed values, \( \hat{x}_i \) is the expected value at time instant \( i \), \( \tilde{x}_i \) is the obtained value at time instant \( i \), i.e., the predicted observation, and \( n \) is the number of predicted observations [45]. Hypothesis tests were also performed on all prediction results, considering the critical value of the statistic \( z = 1.96 \) for the confidence level of 95 percent.

### Table 1

<table>
<thead>
<tr>
<th>pid</th>
<th>inode</th>
<th>amt</th>
<th>time</th>
<th>op</th>
</tr>
</thead>
<tbody>
<tr>
<td>p41</td>
<td>f0</td>
<td>313</td>
<td>24</td>
<td>r</td>
</tr>
<tr>
<td>p89</td>
<td>f3</td>
<td>94</td>
<td>171</td>
<td>w</td>
</tr>
<tr>
<td>p10</td>
<td>f9</td>
<td>92</td>
<td>81</td>
<td>w</td>
</tr>
<tr>
<td>p32</td>
<td>f0</td>
<td>826</td>
<td>132</td>
<td>w</td>
</tr>
<tr>
<td>p69</td>
<td>–</td>
<td>76</td>
<td>70</td>
<td>d</td>
</tr>
<tr>
<td>p1</td>
<td>f5</td>
<td>292</td>
<td>70</td>
<td>w</td>
</tr>
</tbody>
</table>
Finally, our heuristic (Algorithm 1), inspired in [9], is executed to optimize decisions based on predictions, i.e., considering the amount of data to be read or written over time. This heuristic takes decisions on data replication, migration, and consistency. Differently from other approaches that simply consider historical data [9], [16], this heuristic considers the automatic and online prediction of read-and-write operations performed by processes.

Algorithm 1. The proposed heuristic

Require:
Set of sites \( S = \{s_0, s_1, \ldots, s_{n-1}\} \);
Set of process \( P = \{p_0, p_1, \ldots, p_{n-1}\} \);
Set of files \( F = \{f_0, f_1, \ldots, f_{n-1}\} \);
Set of prediction values \( PB \); // in which \( PB \) contains all predicted events based on \( TR \).

\( PB = \{pb_0, pb_1, \ldots, pb_{n-1}\} \);
\( \{ASW\}_p = \) initial value is defined by a constant; // This parameter can be seen as a vector containing the window length of next predictions for every process \( p \in P \), this is why it is indexed by \( p \).
Op = 1; // Number of consecutive homogeneous operations.
\( \beta = 0.10; // \) Adaptive parameter.

Ensure:
Set of replicas \( F' \).

\[
\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2}{x_{\text{max}} - x_{\text{min}}}}.
\]

1: for each \( p \) allocated on grid site \( s \in S \) do
2: for each \( pb \in PB \) do
3: \( p = \) getProcess\( (pb) \)
4: if \( p, \text{op} = \text{"Read"} \) then
5: \( \text{Read}(pb) \)
6: else if \( p, \text{op} = \text{"Write"} \) then
7: \( \text{Invalidate}(f) \)
8: \( \text{Write}(pb) \)
9: else
10: \( PB = \) predict\( (p, \{ASW\}_p, TR) \) // according to Section 3.
11: organize predictions as time series
12: while \( \{ASW\}_p \neq \) Empty do
13: \( \text{Retrieve}(f) \)
14: \( \text{end while} \)
15: \( \{ASW\}_p \leftarrow \{ASW\}_p \times (1 - \beta) + \text{Op} \cdot \frac{\text{Op}}{\text{ASW}} \times \beta \)
16: \( \text{end for} \)
17: \( \text{end for} \)
18: end for

This heuristic maps the grid environment into a graph \( G(S, L) \), with grid sites \( s \in S \) and edges \( l \in L \) represent communication links between elements in \( S \). Every grid site runs a set of processes, previously scheduled according to any policy. Files \( f \in F \) are distributed in sites \( s \in S \). This heuristic also considers \( u \) future observations given by \( PB \), which is the predicted behavior for all processes in \( P \) organized as a sequence of next accesses over time. This algorithm also depends on an initial length for the adaptive sliding window, which estimates the number of observations to be predicted (parameter \( \beta \) is used by this estimator).

An instance of this heuristic is run at every grid site \( s \in S \). It periodically verifies if a process \( p \) will access data, considering predictions \( pb \in PB \). If \( pb \) confirms that \( p \) will access data, then the heuristic needs to define the operation type: read or write (from lines 4 to 8). Otherwise, from lines 12 to 14, the heuristic assesses the feasibility of replication, migration, consistency, and retrieval of files, according to the predicted events, lines 10-11. Line 16 computes the next window length for a process.

Every process \( p \in P \) has its own ASW (represented in Algorithm 1 as a vector of prediction values \( \{ASW\}_p \)). \( PB \) is a subset of the multidimensional time series \( TR \), considering only field \( \text{amt} \) (or amount) in a unidimensional time series. This field was selected due to it provides the amount of data considered in replication, migration, and consistency operations.

The method \( \text{Retrieve}(f) \), Algorithm 2 establishes a new local copy (replication) of the remote file \( f \) and assesses the copy removal option (for data migration purposes). The method \( \text{Read}(pb) \), Algorithm 3 receives a quintuple \( pb \) and, if file \( f \) is \text{invalid} then it applies the method \( \text{Retrieve}(f) \) and finally reads it. The method \( \text{Write}(pb) \) works similarly and is presented in Algorithm 4. The method \( \text{Invalidate}(f) \), Algorithm 5 receives a file and updates all replicas as \text{invalid}.

Algorithm 2. Retrieve\( (f) \)

Require:
The \( f \) file.

Ensure:
\( f' \) replica file retrieved from nearest site.
1: \( A \subset S; // \) subset of sites where there are copies of \( f \).
2: for each \( a \in A \) do
3: \( \text{evaluate costs on replication, migration and consistency} \)
4: \( \text{end for} \)
5: return \( f' \)

Algorithm 3. Read(pb)

Require:
\( pb \).

Ensure:
Reading data from file.
1: if \( f \) is \text{"invalid"} then
2: \( \text{Retrieve}(f) \)
3: \( \text{end if} \)
4: \( \text{reads} \)

Algorithm 4. Write(pb)

Require:
\( pb \).

Ensure:
Writing data to file.
1: if \( f \) is \text{"invalid"} then
2: \( \text{Retrieve}(f) \)
3: \( \text{end if} \)
4: \( \text{writes} \)
In order to validate our approach, we consider some real-world data access optimization results using the proposed technique by Ishii et al. [20] on the three data sets to assess their stochasticity, linearity, and stationarity. After defining such properties for every data set, we then select the most adequate model to represent every one. Recurrence Plot [41] was considered to evaluate stochasticity. White Neural Network [35] was considered to evaluate linearity. Space-Time Separation Plot [47] and Autocorrelation Function (ACF) [48] was considered to evaluate stationarity.

RP evaluates all possible states that can be visited by a series trajectory \( (\tilde{x}_t)_{t=1}^N \), in which \( N \) is the number of considered states in phase space [41], [44]. Moreover, based on RP, Zbilut and Webber [49] propose several measurements (RQA or Recurrence Quantification Analysis) to quantify RP visual structures, such as: the rate or percentage of recurrence (RR), degree of determinism (DET), maximum length of a diagonal line \( (L_{\text{max}}) \), degree of divergence (DIV), and the Shannon Entropy of the probability density function of diagonal lines (ENTR). Such measurements produce a time-dependent behavior which allow the analysis and study of series transitions.

WNN is a test to verify the linearity of time series by using an artificial neural network [35]. The test considers a stochastic process \( Z = \{Z_t, t = 1, \ldots, n\}, Z_t = (Y_i, X_i) \), in which \( Y_i \) is a scalar and \( X_i \) is a vector, and measures the conditional probability, i.e., the probability of \( Y_i \) given \( X_i \), denoted by \( E(Y_i|X_i) \), to conclude about series linearity. Formally, this is represented by a regression function \( g(x) = E(Y_i|X_i) \) which is found after training a multilayer artificial neural network. The objective of that test is to verify if the WNN architecture is capable of mapping function \( g(x) \).

STP [47] is a correlation test to evaluate the geometric trajectories (called global) or fractal structures (called local) of series. STP explicitly considers the time separation of states, i.e., dispersion (distance) between states versus their separation over time. It generates contours that determine the ranges of correlation for a given embedded dimension in a series. This correlation represents the probability that a pair of randomly selected states is lower than a distance \( r \) in phase space [44]. This distance and the separation dimension reflect the position in neighborhood at a given time instant.
ACF [48] measures the degree of correlation among time series observations, i.e., we compute the separation distance of an observation at time instant $t$ against another $t + k$, where $k$ is the time lag. Formally, $ACF(k)$ of a time series $X$ with average $\mu$, is defined in (3), in which $E[.]$ is the expected average value of the expression and $\sigma^2$ is the variance of $X$

$$ACF(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2}, \quad (3)$$

We simply introduce RP, WNN, STP, and ACF to support result analysis. However, we suggest the reading of Ishii et al. [20] in order to better understand such techniques and also issues related to stochasticity, linearity, and stationarity.

In this context, we started assessing the generation process of data set $\text{proj}$, which contains write-only operations. That data set presented the RQA measurements showed in Table 3. By analyzing results, we observe this series presents low recurrence rate and that diagonals confirm it presents low divergence, complexity, and degree of determinism. Furthermore, when analyzing the RP chart (Fig. 4), we confirm the low frequency of diagonals. Given such circumstances, $\text{proj}$ can be classified as a stochastic time series.

Table 4 presents RQA measurements for data set $\text{hm}$, which considers read-only operations. Based on results (Table 4), $\text{hm}$ presents low recurrence rate. Diagonal measurements confirm this series presents moderate divergence, low complexity, and degree of determinism. Moreover, by analyzing the RP chart (Fig. 5), we confirm the low frequency of diagonals. Considering such analysis, $\text{hm}$ can be classified as stochastic.

Table 5 presents RQA measurements for data set $\text{mds}$, which considers read-only operations. By analyzing results (Table 5), we conclude this series presents a very high recurrence rate and the formation of diagonals. Those diagonals confirm this series presents low divergence, moderate complexity, and high degree of determinism. Furthermore, when analyzing the RP chart (Fig. 6), we also confirm the high frequency of diagonals. Given such circumstances, this series is classified as deterministic.

Then, we applied the WNN test [20], [42] to evaluate time series linearity. Table 6 summarizes the WNN test and indicates the classification of the series under study. In this situation, two series are linear and one is not. However, even after applying the linearity test, we still need to identify the degrees of stationarity for each series. Thus, the STP test [20], [38] is applied.

Time series $\text{proj}$ presents the ACF and STP charts presented in Fig. 7. ACF demonstrates $\text{proj}$ has a constant
autocovariance structure which is maintained over time. STP results confirm this autocovariance structure, whose distance (y-axis) values attenuate slowly over time (x-axis). Every curve in STP chart represents a probability \( p \) such as defined in [20], [38]. By analyzing results, we conclude proj is stationary.

Time series \( h_m \) has the ACF and STP charts presented in Fig. 8. The ACF values slowly decay at the beginning and present abrupt changes which are maintained over time, i.e., this does not present a constant autocovariance structure. STP results confirm high values of distance over time. Based on such analysis, we consider \( h_m \) as a nonstationary time series.

Stationarity feature was not evaluated in the \( mds \) time series because previous results conducted on this series allow to classify it as deterministic and nonlinear, as defined in the methodology proposed in [20].

All results previously presented permitted to classify the properties of data sets. Now having such properties, we can better select modeling techniques and, therefore, improve prediction results as seen in [20].

### 5.3 Predicting Process Behaviors

In the next step, data sets were divided in two sets in order to evaluate process behavior predictions: 1) the first with 80 percent of observations for training our approach, and 2) the remainder 20 percent to evaluate predictions results.

Data set proj was the first to be studied. According to our experiments, it was classified as stochastic, linear, and stationary. By considering such classification, this series would be better modeled by using statistic tools, such as AR and ARMA models [20]. This modeling is verified by results presented in Fig. 9, which confirms our hypothesis. In that sense, ARMA(1,2) was capable of modeling proj with the smallest error among all evaluated approaches, indicating good prediction results. ARMA (1,2) was obtained by using the \texttt{auto.arima} tool provided in R statistical software [50], which returns the best order for the arima model to fit data.

Now, consider data set \( h_m \) which is, according to our experiments, classified as stochastic, linear, and nonstationary. According to proposed classification (see Fig. 1) [20], this series would be better modeled by statistic tools, such as ARIMA, which was confirmed by the results presented in Fig. 10. ARIMA(2,1,2) was capable of modeling \( h_m \) with lower errors.

Finally, we consider data set \( mds \) which is, according to experiments, classified as deterministic and nonlinear.
Following the proposed classification (see Fig. 1), this series is better modeled by using chaos-theory or dynamical system tools. Fig. 11 confirms our hypothesis, in which the series is better modeled by using chaos-theory tools (unfolding the behavior in phase space and modeling using RBF). The Phase Space/RBF approach was capable of modeling at lower errors, thus, obtaining good prediction results.

At the end of this stage we have all modeling techniques selected in order to proceed with predictions and the data access optimization.

5.4 Employing the Proposed Optimization Heuristic Considering Predicted Behaviors

All behavior predicted is according to the previous sections is made available to our optimization heuristic, which considers the possibility to anticipate read and write operations as well as consistency verification.

In order to evaluate our proposed heuristic which considers the prediction of process behaviors, we conducted experiments by using the OptorSim simulator, which is part of the European DataGrid EDG Project [13]. This simulator was originally developed to model the dynamic effects of data replication for the Large Hadron Collider Computing Grid (LCG), and it is also considered in other works [9], [14], [15], [16], [17], [18], [19].

Fig. 12 depicts the execution of heuristic in the OptorSim simulator (describes in Section 5.4), considering the online prediction of process behaviors. During execution, our heuristic continuously collects past operations and invokes the Tisean package\(^6\) to conduct predictions according to the modeling techniques selected in the previous phase. In such invocation, the heuristic informs the number of future events to be predicted, which is maintained in the adaptive sliding window (ASW in Algorithm 1). Tisean returns predicted observations to our heuristic which considers such operations to take decisions on replication, migration, and consistency. The adaptive sliding window is modified according to homogeneous or heterogeneous operations, following the idea proposed in [9]. When the length of this window increases, then the heuristic invokes Tisean to predict more observations.

After presenting our proposed heuristic, which optimizes data access operations using online prediction, we conduct several experiments to compare it with other commonly considered techniques. When presenting results, we refer to our heuristic as H-Pred.

We evaluated five optimization techniques: LRU, LFU, Economic Model, H-Hist, and H-Pred (showed in Algorithm 1). LRU replicates files when jobs need them, removing the least-recently-used replicas. LFU replicates files when jobs need them, removing the least-frequently-used replicas. ECO considers an economic model to replicate files. Replicas are removed according to a Zipf distribution.

---

estimation function. H-Hist was previously proposed in [9], and it was considered in this paper for comparative purposes. The first three techniques are available in the default version of the OptorSim simulator. The last two techniques were also implemented in OptorSim.

The first experiment evaluates the performance of data optimization techniques on the \( hm \) data set. The performance metric considered was the average execution time (in seconds). As shown in Fig. 13, both heuristics (H-Hist and H-Pred) were capable of reducing the job execution time in around 60 percent when compared to other data replication techniques. The H-Pred heuristic adopts the ARIMA (2,1,2) model to predict data access operations (see Section 5.2).

The next experiment evaluates the performance of data optimization techniques considering an environment which adopts the \( mds \) data set with read only operations. In Fig. 14, H-Hist and H-Pred heuristics were capable of reducing the job execution time in around 57 percent when compared to other replication techniques. In addition, the H-Pred heuristic adopts the Phase Space/RBF technique to predict data access operations.

The last experiment evaluates the performance of data optimization techniques considering an environment which adopts the \( proj \) data set with write only operations. In Fig. 15, ECO technique has better performance than the H-Hist in around 2 \( \approx 3 \% \). In addition, the H-Pred heuristic, which considered ARMA(1,2) model to predict data access operations, has lower performance than the ECO technique in around 3 \( \approx 4 \% \).

The H-Pred heuristic produces equivalent results or, in some cases (Fig. 15), lower than the H-Hist technique. However, this predictive approach (H-Pred) does not require an initial execution of the application, avoids storage of long historical, and efficiently adapts according to variations on the behavior of applications.

Moreover, it is observed in all charts in this section that, besides increasing the applications performance, historical (H-Hist) and predictive (H-Pred) approaches have low variability in response time when compared to other data optimization techniques, i.e., have low standard deviation. This feature makes both approaches more reliable under different scenarios, which reinforces the use of application knowledge to optimize data access operations.

6 CONCLUDING REMARKS

This paper has presented a data access optimization approach which uses predictive techniques for distributed computing environments. Our main objective is to minimize the application execution time by optimizing data accesses and, therefore, improving decisions on replication, migration, and consistency. From that, data access operations are transformed into time series. By modeling those series, we can understand the behavior of applications and, therefore, predict future observations. Such prediction supports to take decisions beforehand. However, this modeling is related to specific aspects of each time series such as the stochasticity, linearity, and stationarity.

The efficiency of our approach was confirmed through experiments using real-world data. We evaluated our approach to select modeling techniques for real systems data (SNIA data sets [46]). By conducting additional tests, we confirmed that the time series classification can indeed be used as a way to select the most appropriate set of modeling techniques.
Simulations were conducted to study the efficiency of our heuristic which consider predictive mechanisms, under three distinct environments. Experimental results confirm that this approach outperforms other commonly considered ones (e.g., LRU, LFU, and Economic Model) in approximately 50 percent. In addition, our approach is online, avoids the storage of long historical and efficiently adapts according to variations on the behavior of applications.

Acknowledgments
This paper is based upon work supported by FAPESP—São Paulo Research Foundation, Brazil, under grant no. 2011/02655-9, CNPq—National Council for Scientific and Technological Development research funding agency under grants no. 304338/2008-7 and 470739/2008-8. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of FAPESP and CNPq.

References


---

Renato Porfírio Ishii received the PhD degree from the University of São Paulo, São Carlos in 2010. He is currently an assistant professor at the Faculty of Computer Science, Federal University of Mato Grosso do Sul, Campo Grande, Brazil. His research interests include autonomic computing, cluster and grid computing, bioinspired computing, and time series analysis.

Rodrigo Fernandes de Mello received the PhD degree from the University of São Paulo, São Carlos in 2003. He is currently an associate professor at the Department of Computer Science, Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos, Brazil. His research interests include autonomic computing, machine learning, scheduling, and bioinspired computing.

> For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).