Original Article
ISSN (Online): 2582-7472

ShodhKosh: Journal of Visual and Performing Arts
May 2024 5(5), 29–38

THE STUDAY A ORIENTED APPLICATION DATA ACCESS PATTERN ANALYSIS AND PREDICTION

Manmohan [1], Dr. B.D.K. Patro [1]

[1] PhD Scholar, Department Computer Science, Maharishi University of Information Technology, Lucknow, U.P., India

ABSTRACT

An innovative approach for assessing and forecasting object-oriented application runtime behavior in relation to data access patterns across their domain objects is presented in this study. Three different stochastic model implementations are used for the study and forecasts. Markov Chains, Importance Analysis, and Bayesian Inference provide the foundation of the models. The solution handles all required modifications to the target applications under examination in an entirely automated manner, eliminating the need for developer participation. Implementing the TPC-W and oo7 benchmarks validates the findings. Monte Carlo simulations have been used to simulate the oo7 benchmark as a stochas-tic process. It can be shown that our method yields accurate findings in relation to associated with collecting are minimal, between 5% and 9%. A wide range of object-oriented applications may be implemented using the system because to its sufficient flexibility.

Keywords: Data Access Pattern, Persistent Data, Stochastic Modelling, Bayesian Inference, Importance Analysis, Markov Chains, Monte Carlo

Corresponding Author

Manmohan, mohanms75@gmail.com

DOI

10.29121/shodhkosh.v5.i5.2024.1611

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

1. INTRODUCTION

Increases capacity, disc algorithms that translate into an increase in the number of queries that can be handled per second. Maki programmers accountable for carefully optimising an application to minimise the amount of needed satisfy application's the state-of-the-art approach to minimise this issue. But this fine-tuning is challenging, and it often results in an verly lengthy application development process. This approach's additional disadvantage is that structure or altered, this kind of manual tweaking has to be done again. Due to commercial or technological reasons, these modifications are rather typical.

Current three-tier application designs locate the permanent objects management and the business logic on an application server. programmes that use object models that are intrinsically navigational—that is, models in which items

have linkages or references programmes will have difficulties with this design. delay physical disc, applications with navigational features may result in a high number of data-access round trips and unacceptable overhead. The cost of inter-process communication, or this overhead, is growing in significance due to increases handled in a because of advancements capacity, disc algorithms.

Making programmers accountable for carefully optimising an application to minimise the amount of needed to satisfy an application's the state-of-the-art approach to minimise this issue. But this fine-tuning is challenging, and it often results in an overly lengthy application development process. This approach's additional disadvantage is that structure altered, this kind of manual tweaking has to be done again. Due to advancements in technology and commerce, these modifications are rather typical.

The creation of complicated applications is impeded by the existing manual tuning technique, which we contend is unfeasible in the long term. Rather, our goal is to automate the process of tuning by creating self-adjusting techniques for consider. Since that the number of DBMS roundtrips is directly correlated with the elapsed time and these methods aim to reduce it, we anticipate that having these mechanisms in place would reduce the user-perceived elapsed time of jobs. The work presented in this paper focuses on what we believe to be a necessary precondition for mechanisms, plan to address recognise patterns used accurately its future behaviour regarding the domain object manipulations it performs. We specifically outline an automated system that can track an object-oriented application's data access patterns while it runs, analyses those patterns using a variety of models, and forecasts the kinds of domain objects that will likely be accessed when the application is about to run. Many requirements are anticipated to be met by the system, including the following: The system's efficiency lies on its ability to get and evaluate the required statistical data in a way that minimizes overheads for the intended application. Transparency: given the application programmer's perspective, the system ought to be able to accomplish its objectives with little to no assistance from them; Precision refers to how well the system's predictions should match the behaviours that applications have seen in the future.

The article is organised as follows. The associated work is covered in Section 2. We propose a system whose primary structure is described in Section 3. Next, we describe the stochastic models that were used in this study in Section 4, where we first discuss their theoretical underpinnings before going into detail about how they were actually implemented. In Section 5, the outcomes and system assessment are covered. We provide a synopsis and comparative analysis 6, and we examine that our system introduces into the target application in Section 7. In Section 8, we provide last thoughts about our research and the outcomes.

2. RELATED WORK

To prior have used approaches forecast behaviour programmes with regard to the domain data manipulations they execute during runtime, with the exception of the work mentioned in [11, 12]. That being said, there has been some study on the use of stochastic models for telecom traffic analysis that is somewhat related to the methodology presented in this article. We begin by briefly revisiting that work, and then we go on to discuss some of the earlier efforts that we believe might benefit from using the predictions made by the approach we are presenting here to optimise application performance. In [24, 25, 23], a research on the use of stochastic models to telecommunication traffic is described. It has been shown study that several a long tail distribution. When locations distant from the mean or median are given

relatively high probabilities, the probability distribution is said to be long-tailed or heavy-tailed. Note that the present study does not follow the long-tailed assumption that is prevalent in the literature in any way. Nonetheless, these studies are pertinent due to. Specifically, investigated, accounting for both stochastic and deterministic modelling factors, and developing a novel method for traffic behaviour analysis. A flexible method that can accurately model traffic and describe multi-fractal traffic phenomena was also achieved by Li [24] by considering traffic modelling as a generalised Cauchy process.

There are many methods for trying performance optimisations when information about the domain data sets that are most likely to be accommodated is available. Two examples approach, done, are prefetching. Caching is the process of locally caching previously accessed items in memory in order to prevent unnecessary database queries [8].[17] and [28]. While caching has numerous uses, it is ineffective if subsequent requests are seldom identical to the ones that came in first. Prefetching, the process of retrieving data by anticipating an application's future requests, might be useful in these kinds of circumstances [33], [16], and [35]. There may be significant benefits to data-access speed from both prefetching and caching [20] and [1]. Several prefetching experiments, for example, found that prefetching numerous tuples at once, as opposed to just the requested one, improved performance several times [4]. Prefetching methods may be divided into three primary groups [19]: Commercial data servers, object managers, and containers in application servers all utilise deterministic prediction approaches, which follow a fixed pattern. In sequential prefetching, loading one block ahead of time is one example of a scheme [15], [27], and [13]. Pointers from objects to other objects are utilised by object structure-based prefetching algorithms, which are primarily used in OODBMSs to anticipate future data accesses [14], [16], and [4]. Statistical prediction techniques analyse previous accesses to provide probabilistic information about future accesses [31], [22], [10], and [29].

3. SYSTEM DESCRIPTION

Coding injection, data collecting, and data analysis are the three components that make up the system. To insert code into the intended apps, use the code injection module. By rewriting (Java) bytecode, the system carries out this instrumentation at build time. And therefore, without the programmer's help, the system automatically makes all the required changes to the target application. Two instrumentation steps use the Javassist library to insert the code in order to accomplish the modifications. Whaley [34] conducted pertinent research in this area and offered a run-time method for analysing an application's performance as it runs on a Java virtual machine. It is the system's intention to enable a dynamic compiler to make better judgements during code optimisations by providing detailed and continuous system performance information to the compiler. All data accesses occur inside contexts that are identified by the code that is injected during the first instrumentation phase. Among the system's fundamental concepts is context. This specifies the parameters that are used for all data processing. The system allows for several context definitions, however for the sake of the findings reported in this study, the context relates to the manner are made. The target application's execution point may be one of the last n methods calls in the sequence, for example, or the context may relate to the execution of a specific ser- vice. The set of the "possible" contexts would expand e n if a context defined " were used. This is another thing to factor in. In addition to resulting in significantly higher upper memory bounds for the statistical information's storage (explained in more detail

at the end of this section), this would lengthen the amount of time required to gather sufficient statistical data so that the stochastic inference models could produce conclusions. Part 4 will address this last issue in more detail.

Listing 1: Method source code before instrumentation

```
void addPart(Part p) {
 if (this.parts == null) {
  this.parts = new HashSet<Part>();
 }
 p.setCompositePart(this);
 this.parts.add(p)
}
```

The first instruction step modifies each application procedure in order to identify the contexts during runtime. once entering the method, it injects code that changes its related context information, and once returning from it, it cleans up the same context information. Listing 2 shows the final (decompiled) code obtained after the first instrumentation, whereas Listing 1 shows specific procedure prior to this phase's instrumentation. It is important to notice that the renamed Add Part method matches the original method itself. Listing 3 illustrates this.

Listing 2:

Listing 2:

Method source code after first phase

```
void addPart(Part p) { ContextManager.addContextInfo(
new Info("oo7.CompositePart", "addPart"));
renamedAddPart(p); ContextManager.removeContextInfo();
}
```

4. MODELS

The three stochastic models used in the current study are presented in this part with the goal of making predictions about generated. Aside from the real system implementations, features taken into account for each of the models. It should be highlighted that the primary value of this study lies in the manner the models were used to achieve the desired outcome, which is the accurate prediction of object-oriented programmes' behaviour with reference to major data modifications. The models themselves have

4.1. MODEL IMPLEMENTATION

The following section presents the application of the Bayesian Inference Model. We make the forecasts using two kinds of statistical data. The first set, referred known as the "priorset," comprises information regarding historical access patterns certain moment. This relates most recent update of the model forecast. Data from the moment the previous until forecast made is included in the second set, which is referred to as the currentset. It should be mentioned that regular intervals are used to accomplish the model prediction updating. The precise length of these intervals depends on the application and should take a number of things into account. The time spans must to be sufficient to enable the gathering of a substantial amount of behavioural data that is representative. Put another way, the tar-get programme should be granted enough time to complete a sufficient number of operations that correspond to a sample of normal system operations. The length of the time period might range from a few minutes to many hours, depending on the target application

and the amount of work the target system is doing. Two possible issues might arise from the model prediction update at the conclusion of these intervals. The first of these problems is that the model may draw incorrect conclusions about the behaviour of the target system, which was gathered and is used. The second issue is that recalculating the model predictions a lot might result in a performance overhead that could have been avoided.

4.2. THEORETICAL BASE

A method for classifying probable failure modes inside a system according to their severity or impact on the system is called an importance analysis. Throughout the many stages of the system life cycle, it is extensively employed in several industries [9], [18], and [21]. The process via which a failure is noticed and generally explains how it happens is known as the failure mode. evaluating their implications. These techniques are customised for the current job to show which field groupings are more crucial or significant for the functioning of the target application under consideration.

An problem is given a number value in each of three categories: severity (S), occurrence (O), and detection (D) according to the Risk Priority Number (RPN) system, which is a relative grading system. The total RPN for the issue is calculated by multiplying the three ratings together. The specifics of the object under analysis dictate the criteria that are used in each rating system. This figure may be used to rank and compare problems inside the analysis since all issues are scored using the same set of rating scales. It is often inappropriate to compare RPN values across multiple studies, however, since the ratings are given in relation to a specific study.

4.3. RESULTS AND EVALUATION OF THE SYSTEM

We used two distinct benchmarks to assess how well each technique predicted data access patterns in object-oriented applications. The initial benchmark was the TPC-W, which Smith [32] first introduced and which describes mimics operations website for a retail business. The allows simulated to peruse and place product orders. First proposed by Carey [7], the oo7 is the second benchmark. It is often used to evaluate object-oriented persistence techniques' performance. It makes an effort to provide a wide range of functions, enabling the development of an extensive performance profile. Although it does not precisely replicate any one CAD/CAM/CASE application, the oo7 has been developed to have traits that are shared by several of these applications. A number of traversals, modifications, and query operations are carried out across the underlying object model in order to complete the evaluation. Only the results obtained due to the extreme similarity between the two benchmarks' results accuracy made and the performance overheads brought on by the collection of statistical data.

The oo7 benchmark has a few random behavioural aspects, but not enough to demonstrate the viability of a system designed to forecast the behaviour of a target application. Monte Carlo simulations [3] are used to account for it, which causes to act said to stochastic if its behaviour is non-deterministic, meaning that a random element as well as the process's predictable actions determine the system's final state. To do this, a model of the number of methods calls that make up the benchmark is created. Utilising a triangle distribution, the modelling is performed. Because it is the most often used distribution type when working with a system about which little to no information is available on its behaviour, this one was

picked. To mimic the benchmark invocations, three distinct triangular distributions—with left, centre, and right mode locations—are constructed.

4.4. BAYESIAN INFERENCE

In order to create a prediction, the Bayesian inference approach employs two sources of data, as explained in Section 4.1. The present one is the second, while the preceding one is the first. Using the most recent data, the previously gathered information is updated to reflect the trends that have been seen most recently. Both the statistical uncertainty pertaining to the model parameters and the physical uncertainty connected to the variable under consideration may be quantified using the Bayesian technique. One significant characteristic of the Bayesian analysis is the reduction of prediction uncertainty. Figure 2 illustrates this (up). that a falls into is represented, downward). Where p [0,1],. Notably, preceding present represent probability of being read or written belongs to a specific, while represents the variation relative class. This variation is anticipated to be observed in the application's subsequent runs (refer to textbooks on Bayesian statistics, such as [5] and [26]).

The Z test statistic for this type of check can be found in Box [5] as:

$Z=\mu_1 - \mu_2 / \sqrt{\sigma^2/n_1 + \sigma^2/n_2}$ (3) where μ_1 and μ_2 are the mean values, σ^2 and σ^2 are the variances and n_1 and n_2 are the sample sizes, for the predicted and the observed, respectively. This is also known as the decision rule.

Figure 1

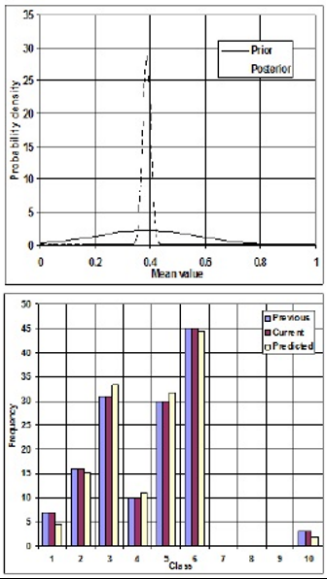

Figure 1 Bayesian Inference, Left Mode Location

The definition of the 1.96, which is equivalent. It is worth mentioning that the level of significance was set at 0.05, which corresponds to the most often used value in engineering practise for these kinds of experiments.

estimated Z values for the three distinct under test, fall within the 95% confidence level. This indicates that the mean values of the expected and subsequently observed (patterns) do not vary significantly at the 0.05 level of

significance. As a result, we can affirm that the system's predictions are accurate. They are able to accurately identify the patterns that are really seen in the application's subsequent executions.

4.5. IMPORTANCE ANALYSIS

The findings from using the Importance Analysis Model on the will shown. The criticality rank table below should be carefully considered before the actual findings are shown.

Table 1

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Table 1 Criticality Rank Table

Tables like serve when deciding particular regarded, depending on. This is explained theoretical section of the Importance Analysis Model., as it relates to the present work, corresponds to the field of a certain application class, and its significance or criticality is linked to the likelihood that the application would access that field in a particular scenario. Therefore, the field is deemed very significant for the functioning of the application if, particular corresponds those in dark grey region. In contrast, it is quite improbable that the application would need a field if its location is within the non-shaded region. The remaining fields in the intermediate region are considered to be of ordinary significance.

It is noteworthy that a significant portion of the application areas fall into the lower probability groups, with just a small minority falling into the higher probability classes. The benefits resulting from this trend are similar to those found in the findings of the Bayesian Inference model; that is, performing optimisations on a smaller collection of data that is more likely to be required is simpler

Figure 3

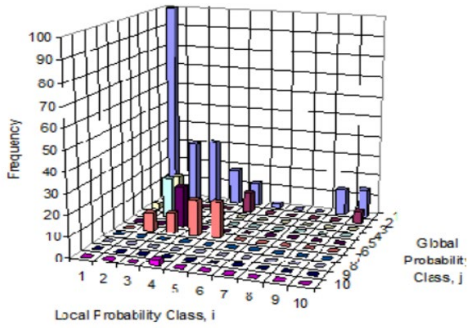

Figure 3 Criticality, left Mode Location

4.6. MODEL EVALUATION

One may conduct a comparative analysis based on the attributes shown by the three unique models created in this study. The most intricate of the three models, in terms of semantic complexity, is the one based on Bayesian inference. It is most straightforward importance. Note that simple in practise since this comparison is based on a relative scale. What comes next is the analysis of each model's output stability with respect to the variations in the target application's behaviour. The Markov Chain has been shown to be the most stable model of all; the Importance Analysis model shows an intermediate level of stability, while the model to be susceptible differences.

Lastly, three ' outputs because the models used differ significantly from one another. The only way to compare the models' outputs effectively is to use some sort of optimisation technique, like guided prefetch or caching policies. Despite this, a similar pattern about falling into can be seen in the outcomes of the three new models that were created here. Only a small percentage of the application fields in all three models are in higher classes of actual likelihood. The fields that are across.

4.7. OVERHEAD ANALYSIS

The application has a substantial that is to gather all data required construct the models. When compared to the application's non-instrumented version, the need to run this code results in overruns and negatively impacts performance. It is crucial to quantify such overheads in order to ascertain whether or not the application's regular operations can tolerate them. An additional factor to be considered is the amount of memory required to store collected certain.

The methods used to collect vary significantly used hence, individual will be examined independently. Moreover Finally, it should be mentioned that, in contrast to the previous two analysis models, which are expected to have uninterrupted data acquisition mechanisms while the application is functioning normally, collects data model's phase. Due to the expectation that the Markov chain data collection model would continue to acquire data throughout the regular course of the application, the restrictions surrounding the overheads imposed by this model are therefore significantly less stringent than those for the other two models.

Next, we give Overhead, computed follows:

$$\overline{overhead} = \frac{\sum_{i=1}^{n} overhead_{method,i} \times t_{method,i}}{\sum_{i=1}^{n} t_{method,i}} \quad (4)$$

When t is the execution time and the expressed as a the. The performance overhead's weighted average is 5.15%. This is why, as compared to the original form of executes around 5% slower on average. This performance penalty is thought to be reasonable. Concerning the examination of overheads brought about by the collection of data overhead weighted average in this instance, as determined by Equation (4), is 9.14%. As a result, the tar-get programme will operate, counterpart when it is in training mode, collecting. Regarding the extra memory requirements, it is impossible to confirm if there is a " between requirements of using created because of the storing of on obstructed access patterns. An real example would be to state that, while the benchmark process would demand several hundred MB of

memory to execute, the statistical data collected from several benchmark executions would not surpass few hundred KB when stored on the hard drive. This leads us to the conclusion that the system's memory needs are minimal for storing the statistical data required for the creation of forecasts.

5. CONCLUSIONS

The discussed created that analyses forecast likely patterns that are made over domain data while object-oriented programmes are being executed. The TPC-W and oo7 benchmarks were run in order to verify the system's accurate and proper operation. The three stochastic models that are used in the system are Markov Chains, Importance Analysis, and Bayesian Inference. They are used to produce predictions about the patterns of data access that target applications are most likely to follow while they are running. A number of unique benchmark examples were examined and replicated. All of the models that were used produced satisfactory results. automatically applications obtain the statistical data that is utilised as input for the models. The data collection procedure introduces overheads into the target applications, which vary from 5% for the Markov Chains to 9%. These overheads are deemed acceptable.

It has been shown that the approach created with this study is sufficiently adaptable to be used with any object-oriented application. We have designed a new technology that can be used to many target applications without requiring any special adjustments.

CONFLICT OF INTERESTS

None.

ACKNOWLEDGMENTS

None.

REFERENCES

Adali, S., Candan, K. S., Papakonstantinou, Y., and Subrahmanian, V. S. Query caching and optimization in distributed mediator systems.SIG- MOD Rec., 25:137–146, June 1996

Aitchison, J. and Dunsmore, I.Statistical Prediction Analysis. Cambridge University Press, New York, NY, USA, 1975

Berg, B.Markov Chain Monte Carlo Simulations and Their Statistical Analysis. World Scientific, 2004.

Bernstein, P. A., Pal, S., and Shutt, D. Context- based prefetch an optimization for implementing objects on relations. The VLDB Journal, 9:177– 189, December 2000.

Box, G. and Tiao, G.Bayesian Inference in Sta tis- tical Analysis. Wiley, New York, NY, USA, 1992.

Brin, S. and Page, L. The anatomy of a large-scale hypertextual web search engine.Comput. Netw. ISDN Syst., 30:107–117, April 1998.

Carey, M., Dewitt, D., and Naughton, J. The oo7 benchmark. In Proceedings of the ACM SIG- MOD International Conference on Management of Data, pages 12–21, 1993.

Dar, S., Franklin, M. J., Jónsson, B. T., Srivastava, D., and Tan, M. Semantic data caching and replacement. In Proceedings of the 22th Inter- national Conference on

Very Large Data Bases, VLDB '96, pages 330–341, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

DoD, U. S. Procedures for performing a failure mode effects and criticality analysis. MIL-STD- 1629A, 1984.

Drapeau, S., Roncancio, C., and Guerrero, E. Generating association rules for prefetching. In Proceedings of the ICDCS Workshop of Knowledge Discovery and Data Mining in the World- Wide Web, pages F15–F22, Taipei, Taiwan, 2000