

Mining Hidden Mixture Context With ADIOS-P To Improve Predictive Pre-fetcher Accuracy

Jong Youl Choi*, Hasan Abbasi*, David Pugmire*, Norbert Podhorszki*, Scott Klasky*, Cristian Capdevila†, Manish Parashar‡, Matthew Wolf§, Judy Qiu¶ and Geoffrey Fox¶

* Scientific Data Group, Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA

† Electrical Engineering and Computer Science, The University of Tennessee, Knoxville, Tennessee, USA

‡ Electrical and Computer Engineering, Rutgers University, Piscataway, New Jersey, USA

§ School of Computer Science, Georgia Institute of Technology, Atlanta, Georgia, USA

¶ School of Informatics and Computing, Indiana University, Bloomington, Indiana, USA

{choij, habbasi, pugmire, pnorbert, klasky}@ornl.gov,

ccapdevi@utk.edu, parashar@rutgers.edu, mwolf@cc.gatech.edu, {xqiu, gcf}@indiana.edu

Abstract—Predictive pre-fetcher, which predicts future data access events and loads the data before users requests, has been widely studied, especially in file systems or web contents servers, to reduce data load latency. Especially in scientific data visualization, pre-fetching can reduce the IO waiting time.

In order to increase the accuracy, we apply a data mining technique to extract hidden information. More specifically, we apply a data mining technique for discovering the hidden contexts in data access patterns and make prediction based on the inferred context to boost the accuracy. In particular, we performed Probabilistic Latent Semantic Analysis (PLSA), a mixture model based algorithm popular in the text mining area, to mine hidden contexts from the collected user access patterns and, then, we run a predictor within the discovered context. We further improve PLSA by applying the Deterministic Annealing (DA) method to overcome the local optimum problem.

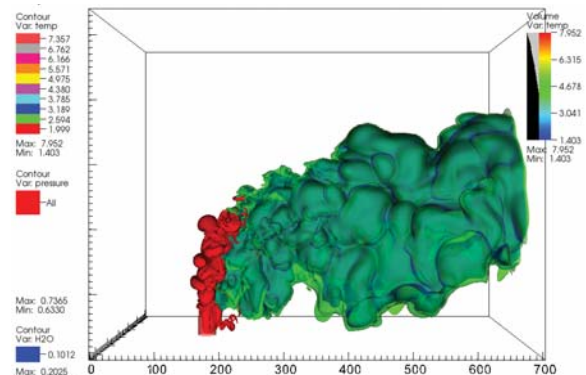
In this paper we demonstrate how we can apply PLSA and DA optimization to mine hidden contexts from users data access patterns and improve predictive pre-fetcher performance.

Index Terms—prefetch; hidden context mining;

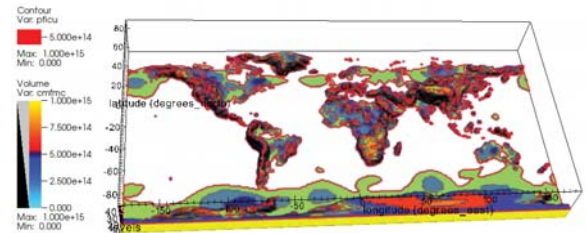
I. INTRODUCTION

Over the past decade the computing industry in general, and the HPC community in particular, has seen an explosive growth in computing power, driven primarily by the industry’s need to keep up with Moore’s law. This has resulted in the Top500 moving from almost 5 TFlops in the year 2000, to more than 16 PFlops in 2012, an astounding increase of 3 orders of magnitude [1, 2]! This substantial growth in computing power in FLOPs has far outpaced growth in other aspects of computing, particularly in the area of IO-related technologies. Today, IO has become a significant source of performance bottleneck for scientific applications.

In this era of data explosion, deploying a predictive data pre-fetcher has been considered as a viable solution to load data in before real requests happen, i.e., if one can predict incoming data access patterns, the data can be pre-fetched or pre-loaded to reduce data loading latency. This idea is not brand-new and has been explored for years in the area of serving shared resources. During the development and deployment of our IO middleware, Adaptive IO System (ADIOS) [3], we



(a) Combustion simulation by S3D



(b) Climate simulation output from GEOS-5

Fig. 1: Examples of visualization of S3D (a) and GEOS-5 (b) with multiple variables through VisIt.

have observed the potential for pre-fetching in many real-world large-scale scientific applications, such as combustion simulation (S3D) [4], climate modeling (GEOS-5) [5], Gyrokinetic Toroidal Code (GTC) [6], plasma fusion simulation code (XGC) [7], amongst others, as well as for parallel visualization softwares for scientific data, such as VisIt (examples are shown in Fig. 1). ADIOS is designed to improve IO performance by orchestrating various types of IO requests transparently, targeting large-scale and data-intensive scientific applications. ADIOS has been designed as an extensible middleware, and we have taken advantage of this characteristic to add a provenance module, called ADIOS-P, by which file-related activities will be stored, indexed, and queried later (Fig. 2).

In a large-scale scientific data visualization, scientists often want to compare multiple variables embedded in a single

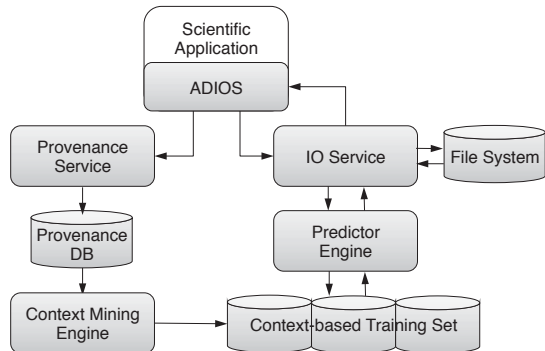


Fig. 2: Overview of the ADIOS Provenance system, ADIOS-P

file (like HDF5 or NetCDF) or spanned from multiple files, by rendering them together in an interactive way (i.e., the sequences of loading variables are somehow spontaneous). However, most visualization operations are data-intensive, meaning IO operations consume the majority of the time [8]. Users will waste most of their time waiting for the completion of IO requests. In this scenario accurately coordinated data pre-fetching may help to significantly reduce the IO time by loading data into the memory of the visualization software while users are sitting idle or manipulating graphic objects, operations without substantial IO requirements.

Our goal is to utilize collected history of file or variable access activities for developing a predictor that will discover patterns of file accesses and be able to forecast the upcoming file access requests. With this predictor, we will be able to proactively pre-fetch the data that are likely to be requested and thus expect to reduce IO latency in scientific applications.

As mentioned, extensive research on developing a predictor have been performed in the file system areas and various types of algorithms have been proposed to mine underlying correlation between accessed files; frequent set mining, network models, Markov chain model. However, not many researches have discussed the importance of preparing training set. In this paper we focus how we can improve predictor accuracy by preparing a training set with an informed way.

Our problem is shown in Fig. 3. In many data file access patterns (in scientific applications) or variable access patterns (in visualization software) we have observed, a user accesses multiple files (or variables) with different purposes within a session. For example, one can open a sequence of files for data analysis and visualization and other files for writing reports. Some of them can be opened for both purposes. We call those purposes *contexts*. In general, contexts are hidden as they are not explicitly exposed in the access logs or traces from which we build a predictor. The intuition is that if we discover a users' intentions or contexts, we can build a better predictor, i.e., if we train a predictor in a more informed manner by using context-aware training, we can improve its accuracy. This concept is inspired from the text mining algorithms based on the topic model in which the purpose is to discover hidden topics (or contexts) and model documents as a mixture of multiple topics. In our case, we model file (or variable) access

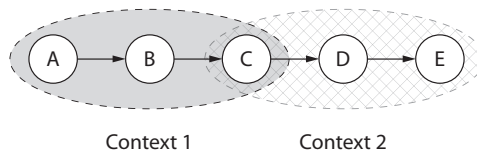


Fig. 3: An example of a mixture of contexts. A session consists of two contexts which share item C together. C is called polysemous or multi-contextual.

patterns as a mixture of contexts.

For mining hidden contexts, we used the Probabilistic Latent Semantic Analysis (PLSA) algorithm [9, 10], derived from a mixture model [11]. In the text mining area, PLSA has been popularly used for building a probabilistic model for languages and documents posing the problems of synonymy (different words sharing a same concept) and polysemy (a same word having different meanings), from which we also suffer in analyzing file access patterns, i.e., there are multiple files used in a same context or the same file used in different multiple contexts. We observed simply applying the PLSA algorithm itself can improve prediction accuracy (More details will be discussed in Section V). However, we take a further step toward squeezing the prediction quality by improving the PLSA algorithm. PLSA is natively suffered from the local optimum problem because its optimization routine is based on the Expectation-Maximization (EM) algorithm [12]. We find a more optimized solution by using the Deterministic Annealing (DA) method to improve prediction quality and accuracy.

Our contribution in this paper is summarized as follows:

- Propose a hidden context mining algorithm to train predictive predictors built around the ADIOS provenance system, called ADIOS-P.
- Demonstrate experimental results showing improvements in prefetching accuracy and data read performance by using two trace data sets; DFSTrace file access traces [13] and variable access logs collected from VisIt through the ADIOS provenance system, ADIOS-P.
- Propose a pre-fetcher performance model from which we can estimate an improved IO throughput.

II. BACKGROUND

A. Predictive Prefetching

Developing predictive algorithms for the purpose of pre-fetching has been extensively studied in the areas of using shared resources, such as file systems, metadata services, web services, etc. In our paper we focus on mining IO access patterns from the logs of file or variable accesses.

Formally, we define file access patterns as follows (similar analogy can be made for variable access patterns). Assume we have a total L files $\{x_1, \dots, x_L\}$ in the system. During the i -th session s_i , a user accesses a sequence of M_i files, $(a_1, a_2, \dots, a_{M_i})$, where access a_j corresponds to a file among L files. Then, we denote the history, or the collection of sessions, as $H = \{s_1, \dots, s_N\}$. The purpose of prediction is to forecast the next upcoming file access in a given session based on the history H .

In a graphical model, file access patterns are summarized as a directed graph in which each node represents a file and an edge between two nodes, say, x_i and x_j , represents a conditional probability $P(x_j|x_i)$ meaning the probability of file x_j accessed after file x_i . This model is also known as a Markov chain describing file accesses activities as a finite state transition. If we consider consecutive \mathcal{N} transitions to compute the probability of file x_j access, we can build a \mathcal{N} -th order Markov chain in which the probability can be represented by $P(x_j|x_{j-1}, \dots, x_{j-\mathcal{N}})$.

Nexus [14, 15], another prefetching algorithm based on a graph model with weighted edges, has been proposed. Nexus is a variant of a \mathcal{N} -th order Markov chain with a decaying effect in a way in which the conditional probability between two nodes is decreasing as the path length is getting larger. Please refer to the original papers [14, 15] for more details of the algorithms.

B. Probabilistic Latent Semantic Analysis (PLSA)

PLSA [9, 10] is an algorithm seeking a generative process of observed data, from which one can discover essential probabilistic structures or latent aspects of data. Most notably, PLSA is one of the most used algorithms applied in analyzing and retrieval of text document [11, 16]. PLSA originally stemmed from Latent Semantic Analysis (LSA) [17, 18], a method to summarize data based on a linear combination of L_2 -norm approximation and provides a principled approach to build a statistical model of data.

In a nutshell, PLSA is based on a latent mixture model, in which data (or *documents*) is represented by a mixture of finite number of latent components (or *topics*). In other words, PLSA seeks a finite number of topics, say K topics, which can represent optimally the group of documents.

In PLSA, we denote a collection of N text documents, called a *corpus*, as $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ where \mathbf{x}_i ($1 \leq i \leq N$) represents a document vector. In this corpus, we have a vocabulary set containing D unique words (or *terms*) denoted by $\{w_1, \dots, w_D\}$ and each document \mathbf{x}_i is a D -dimensional vector where its j -th element represents the number of occurrences (or *frequency*) of word w_j . One may summarize the corpus \mathbf{X} in a rectangular $N \times D$ matrix, called co-occurrence (or document-term) matrix $\mathbf{X} = [x_{ij}]_{ij}$ for $1 \leq i \leq N$ and $1 \leq j \leq D$, in a way in which an element x_{ij} denotes the frequency of word w_j occurred in a document \mathbf{x}_i . In this paper, we use PLSA to analyze the trace data for files or variable access logs, in which we can translate sessions as documents and words as file names or variable names in PLSA.

Then, we define a topic as a generative function that will create a document (i.e, a list of words and word frequencies) with a multinomial distribution over words. More specifically, if a document is generated from a certain topic, say k -th topic, its conditional probability can be written by

$$P(\mathbf{x}_i | \zeta_k = 1) = \text{Multi}(\mathbf{x}_i | \boldsymbol{\theta}_k) \quad (1)$$

where ζ_k is called a latent class, a binary random variable indicating association with the k -th latent class, and

$\text{Multi}(\mathbf{x}_i | \boldsymbol{\theta}_k)$ represents a multinomial probability of \mathbf{x}_i over word probability $\boldsymbol{\theta}_k = (\theta_{k1}, \dots, \theta_{kD})$ where θ_{kj} represents a word probability $P(w_j | \zeta_k = 1)$, defined by

$$\text{Multi}(\mathbf{x}_i | \boldsymbol{\theta}_k) = \frac{\Gamma(|\mathbf{x}_i| + 1)}{\prod_{j=1}^D \Gamma(x_{ij} + 1)} \prod_{j=1}^D (\theta_{kj})^{x_{ij}} \quad (2)$$

with a gamma function, $\Gamma(\cdot)$.

Assuming we have total K topics in a given corpus, the marginal document probability can be defined as a mixture of topics written by

$$P(\mathbf{x}_i | \boldsymbol{\Theta}, \boldsymbol{\Psi}) = \sum_{k=1}^K \psi_{ik} \text{Multi}(\mathbf{x}_i | \boldsymbol{\theta}_k) \quad (3)$$

where a word probability set is denoted by $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K\}$ and a mixture weight set is presented by $\boldsymbol{\Psi} = [\psi_{ik}]_{ik}$ for each mixture weight ψ_{ik} with the constraint $0 \leq \psi_{ik} \leq 1$ and $\sum_k \psi_{ik} = 1$. Note that a mixture weight ψ_{ik} is a document level parameter, rather than a corpus level, in that each document can have different mixture weights over the finite number of topics. This is the key difference between clustering algorithms, like K-Means, and the topic model.

Then, PLSA is a problem to seek an optimal set of parameters which maximizing the log-likelihood defined by

$$\mathcal{L}_{PLSA}(\mathbf{X}, \boldsymbol{\Theta}, \boldsymbol{\Psi}) = \sum_{i=1}^N \log \left\{ \sum_{k=1}^K \psi_{ik} \text{Multi}(\mathbf{x}_i | \boldsymbol{\theta}_k) \right\}. \quad (4)$$

Finding such parameters in this mixture model, known as model fitting or parameter estimation, is intractable. The original PLSA algorithm maximizes the objective function (4) by using the Expectation Maximization (EM) method.

In Section IV we will discuss how the Deterministic Annealing (DA) algorithm can be used to get better optimized solution for the PLSA problem.

III. RELATED WORK

We discuss related previous research on predictive prefetching and deterministic annealing.

Predictive Prefetching: Predictive prefetching has been widely studied in the areas of file system and web services to reduce the file loading or web page access time. In the file system area, a series of Partitioned Context Modeling (PCM) based schemes [19–21] have been studied for sequence file prediction for IO prefetching. AMP [22] and TaP [23] have been proposed for sequence prediction. Researches about the prefetching in shared file storage [15, 24] have been performed. Memory prefetching schemes [25, 26] have been also researched to increase cache performance.

In the web service area, G. Pallis et al. proposed clustWeb, a graph-based clustering algorithm for web pages, and clustPref, a web prefetching scheme based on clustWeb algorithm, to improve network performance by using a predictive approach and reported a significant performance improvement [27]. We take a similar clustering-based approach but we focus on a mixture model based algorithm for file systems in which mining hidden users contexts or intensions are important.

Deterministic Annealing (DA): The DA algorithm [28, 29] has been applied to solve optimization problems in various machine learning algorithms, such as clustering [28, 30, 31], visualization [32, 33], protein alignment [34], and so on. A good modern DA overview can be found in [35]. A general DA solution for EM algorithm is proposed in [12].

Our focus in this paper is to solve a EM-based text mining algorithm, PLSA, by using DA. T. Hofmann, the author of the PLSA algorithm, has also proposed a DA-like algorithm, called Tempered EM [9]. However, the Tempered EM is different from the traditional DA algorithm in that the cooling schedule is reversed and is only applied to solve overfitting problem. Our proposed algorithm, Probabilistic Latent Semantic Analysis with Deterministic Annealing (DA-PLSA), is closer to the original DA approach presented by K. Rose and G. Fox [28, 29].

IV. MINING HIDDEN MIXTURE CONTEXT WITH DETERMINISTIC ANNEALING

To maximize the log-likelihood function shown in Eq. (4), T. Hofmann has proposed an EM algorithm for model fitting in PLSA [9, 10]. However, EM has a well-known problem, called a local optimum problem, finding only local solutions. To overcome such problem, we propose a new DA algorithm for PLSA, named Probabilistic Latent Semantic Analysis with Deterministic Annealing (DA-PLSA). We follow the same approach in solving a clustering problem with DA presented by K. Rose and G. Fox [28, 29].

The DA algorithm, based on the principle of maximum entropy [36], developed by E. T. Jaynes, a rational approach to choose the most unbiased and non-committal answer for a given condition, was developed to avoid local optimum and seek a global optimum solution in a deterministic way [28], which contrasts to stochastic methods used in the simulated annealing [37], by controlling the level of randomness or smoothness. The DA algorithm, adapted from a physical process known as annealing, finds an optimal solution in a way gradually lowering a numeric *temperature* which controls randomness or smoothness

In DA, we optimize a new objective function \mathcal{F} , called *free energy*, similar to the Helmholtz free energy in statistical physics, defined by

$$\mathcal{F} = \langle \mathcal{D} \rangle - T \mathcal{S} \quad (5)$$

where $\langle \mathcal{D} \rangle$ represents an expected cost, T is a Lagrange multiplier, also known as a *numeric temperature*, and \mathcal{S} is an entropy.

To solve the PLSA problem with DA, we define the following objective function, *free energy* \mathcal{F}_{PLSA} , by

$$\mathcal{F}_{PLSA} = -\frac{1}{\beta} \sum_{i=1}^N \log \sum_{k=1}^K \{\psi_{ki} \text{Multi}(\mathbf{x}_i | \boldsymbol{\theta}_k)\}^\beta \quad (6)$$

where β represents *inverse computational temperature*, defined by $\beta = 1/T$. Please note that the free energy function (6) equals with the EM objective function (4) when temperature

TABLE I: List of trace data sets

| Data Set | Name | Users | Sessions | Files/Variables |
|---------------|--------|-------|----------|-----------------|
| DFSTrace | barber | 9 | 6,410 | 6,833 |
| | dvorak | 8 | 29,613 | 16,815 |
| | ives | 12 | 7,724 | 5,776 |
| | mozart | 10 | 11,026 | 13,687 |
| VisIt+ADIOS-P | VisIt | | 27 | 94 |

is 1.0, which implies that the DA algorithm treats the EM solution as a special case.

With Eq. (6), we will gradually lower a temperature from high to low (equivalently, β will be changed from near zero to 1). At each temperature, we have the following internal EM steps to minimize \mathcal{F}_{PLSA} .

- E-step : compute ρ_{ki} , known as the responsibility, by

$$\rho_{ki} = \frac{\{\psi_{ki} \text{Multi}(\mathbf{x}_i | \boldsymbol{\theta}_k)\}^\beta}{\sum_{k'=1}^K \{\psi_{k'i} \text{Multi}(\mathbf{x}_i | \boldsymbol{\theta}_{k'})\}^\beta} \quad (7)$$

- M-step : maximize \mathcal{F}_{PLSA} by computing the following parameters:

$$\boldsymbol{\theta}_k = \frac{\sum_{n=1}^N \rho_{ki} \mathbf{x}_i}{\sum_{n=1}^N \rho_{ki}}, \quad \psi_{ik} = \frac{\rho_{ki}}{\sum_{k=1}^K \rho_{ki}} \quad (8)$$

which make the first derivative of \mathcal{F}_{PLSA} , $\partial \mathcal{F}_{PLSA} / \partial \boldsymbol{\theta}_k$ and $\partial \mathcal{F}_{PLSA} / \partial \psi_{ik}$, be zero.

In summary, with DA-PLSA, we model the collection of sessions H , defined in II-A; i) each session can be described as a mixture of hidden K contexts with different mixing weights, and ii) K generative processes can be inferred. In conjunction with a predictor, we can exploit those properties to increase the prediction performance; prediction will be made based on the major context or multiple contexts to which each session belongs.

V. EXPERIMENTAL RESULT

In this section, we demonstrate our experimental results evaluating the impact of using hidden context mining algorithms to improve the predictor performance. As our focus is mining hidden contexts for training predictive pre-fetcher, we chose two basic and popular predictors, \mathcal{N} -th order Markov chain (nMarkov hereafter) and Nexus.

For the trace data, we have used two datasets. First, a publicly available file trace data, called DFSTrace [13], generated from the Coda project [38]. Second, a dataset collected directly from the visualization software, VisIt, integrated with the ADIOS provenance module, ADIOS-P. The DFSTrace consists of 4 different datasets collected from the different machines, *barber*, *ives*, *dvorak*, and *mozart*, each of which has unique file access characteristics [19]; *barber* has the highest rate of system calls per second, *dvorak* has the highest percentage of write activity, *ives* the largest number of users, and *mozart* a typical desktop work-station.

To prepare the VisIt trace data, we integrated the VisIt program with the ADIOS-P module and recorded variable access activities while users' performing visualization of outputs

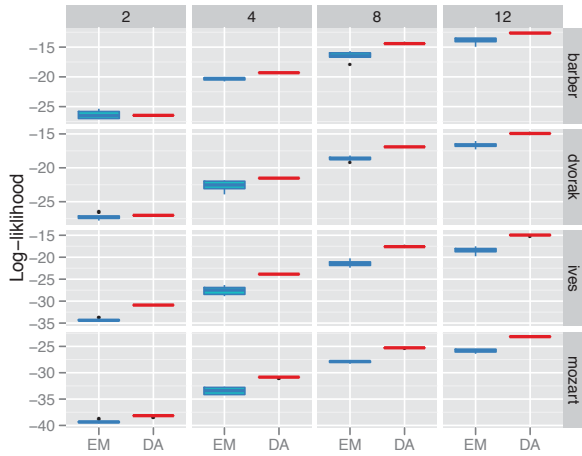


Fig. 4: EM-optimized vs DA-optimized PLSA. DA optimization outperforms EM-optimization by showing larger log-likelihood values than EM.

from the real scientific applications, such as S3D [4], GEOS-5 [5], etc. Examples of S3D and GEOS-5 outputs are shown in Fig. 1. The trace data used in this paper is summarized in Table I.

A. Deterministic Annealing performance

We compare the optimization performance of the original, EM-optimized PLSA algorithm (EM-PLSA) with the one with our proposed DA-optimized PLSA algorithm (DA-PLSA). We performed the EM-PLSA and DA-PLSA algorithms by using the DFSTrace data sets while using different context numbers, $K = 2, 4, 8$, and 12 , and measured the maximum log-likelihood values as outputs. Models with larger log-likelihood values are preferred. We repeated the process with 10 randomly initialized conditions.

The results are summarized in Fig. 4 by using box plots (also known as a box-and-whisker diagram) in which observed data are represented by a box for the upper quartile (Q3) and lower quartile (Q1), whiskers for the smallest and the largest values, a middle line in the box for the median, and

As a result (Fig. 4), the DA optimization outperformed the EM method by generating larger log-likelihood values than EM in all the cases we tested, except *barber* with $K = 2$. On average, the log-likelihood values generated by DA-PLSA were bigger by 2.028 than ones from EM-PLSA.

Please note also that the variance of log-likelihood values generated from DA-PLSA is smaller than the one from EM-PLSA; the average standard deviations of log-likelihood are 0.029 and 0.510 for DA-PLSA and EM-PLSA respectively. This illustrates the robustness of our DA method; DA-PLSA is less sensitive to random initial conditions than EM-PLSA. In short, this experiment demonstrates that our DA-PLSA algorithm finds better model parameters than EM-PLSA with smaller deviations.

B. Impacts on prediction quality

In this experiment, we measure how context mining algorithms can improve prediction quality. We used the following

4 context mining algorithms:

- K-Means – algorithm to find K clusters based on Euclidean distance measures. Strictly speaking, K-Means is not considered as a topic or context mining algorithm in general. We used as a simple initial approach.
- PLSA – The original, EM-optimized algorithm.
- DA-PLSA – Our proposed, DA-optimized algorithm.
- DA-PLSA2 – The DA-PLSA algorithm with 2 modes (explained below).

For PLSA family algorithms, we limited the mode, the maximum number of membership that each session can be associated with, be 1, except DA-PLSA2 which was set to have 2 modes; in PLSA and DA-PLSA, each session will be associated with one latent context group, while in DA-PLSA2 two latent groups will be selected for each session.

Then, we measured the impacts of using the 4 different context mining algorithms on the prediction performances of two sequence predictors, Nexus and nMarkov, by using two trace datasets; the DFSTrace datasets (*barber*, *ives*, *dvorak*, and *mozart*) and the VisIt trace set. More specifically, first, we measured the prediction accuracy of Nexus and nMarkov without using any context information. This measurement was used as base values. Then, we performed the prediction again by using the context information mined from K-Means, PLSA, DA-PLSA, and DA-PLSA2 and measured the prediction accuracies. Then, we calculated the percentage differences compared with the base values. Positive percentage difference values indicate the performance (or prediction accuracy) improvements.

The experimental results with the DFSTrace sets are shown in Fig. 5 by using a box plot (We omit the results of *mozart* due to the space limitation. However, the results were similar with others). As a result, DA-optimized PLSA algorithms (DA-PLSA and DA-PLSA2) outperformed the original, EM-optimized algorithm (PLSA) and K-Means. Especially, DA-PLSA with 2 mode membership (DA-PLSA2) showed the best performances in all the cases by generating the largest accuracy improvements. Most notably, DA-PLSA2 in *dvorak* showed about 90% prediction accuracy increase with Nexus at $K=8$ and 12 .

However, for small K values ($K=2$ or 4), the performance improvement was not impressive (although DA-PLSA2 still worked better than any others in most cases). This is expected as mining of small number of hidden contexts is not much different from clustering and there is not much model-specific information we can exploit to improve prediction quality.

Fig. 6 shows another experimental results with the VisIt trace data, containing the variable access history. However, due to the limited time of collecting trace data, the data set contains only the small number of sessions. By using the same method to compute base values (prediction accuracy without using context mining results), we measured the percentage improvements of prediction algorithm by using the context mining results. In Fig. 6, we can see also DA-PLSA2 outperformed than any other algorithms, K-Means, PLSA, and DA-PLSA. However, unlike in the previous experiment, we

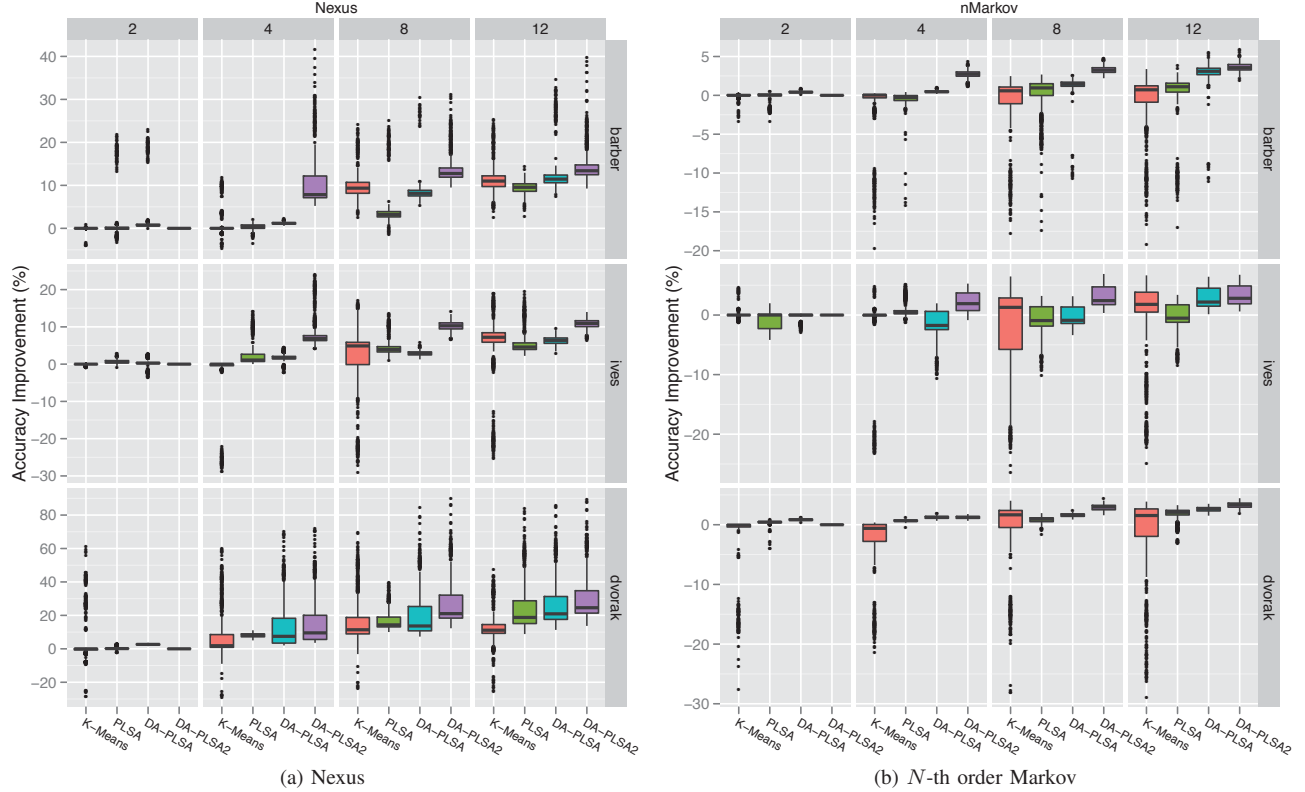


Fig. 5: Impacts of context mining algorithms on the prediction performance of Nexus (a) and nMarkov (b) for DFSTrace data, depicted as box plots. For each box plot, observed data are summarized as a box, whiskers, and circles; top of box for the upper quartile (Q3), bottom of box for the lower quartile (Q1), whiskers for the smallest and the largest values, a middle line in the box for the median, and circles for outliers. DA-optimized PLSA algorithms (DA-PLSA and DA-PLSA2) outperform the original, EM-optimized PLSA algorithm. Especially DA-PLSA with 2 mode membership (DA-PLSA2) shows the best performances overall.

observed the poor performance of DA-PLSA. We think this is due to the small number of trace events.

C. IO pre-fetch performance

We demonstrate how prefetching can affect IO performance. Instead of measuring IO performance directly in a system with prefetching deployed, we performed a simulation-based experiment by building a performance model.

We consider a 1:N parallel execution model in which there are 1 staging process (P_s) mainly for IO handling and prefetching and N computing processes (P_n) which will communicate with the staging process P_s for IOs (See Fig. 7). For simplicity, we ignore communication overheads between processes in this model. We also assume an iterative map-reduce style execution [39]; a full execution can be divided into small sub steps and a computing task in each step begins with an IO input received from the staging process P_s and ends in a synchronized fashion so that the next IO reading follows after the end of the computing. The staging process P_s handles IO sequentially, while N computing processes, P_1, \dots, P_N , can run concurrently.

We denote $T(r_t^i)$ as an elapsed time for reading t -step data for process P_i and $T(c_t, P)$ as a parallel computing time, or longest computing time among N processes. Then, we

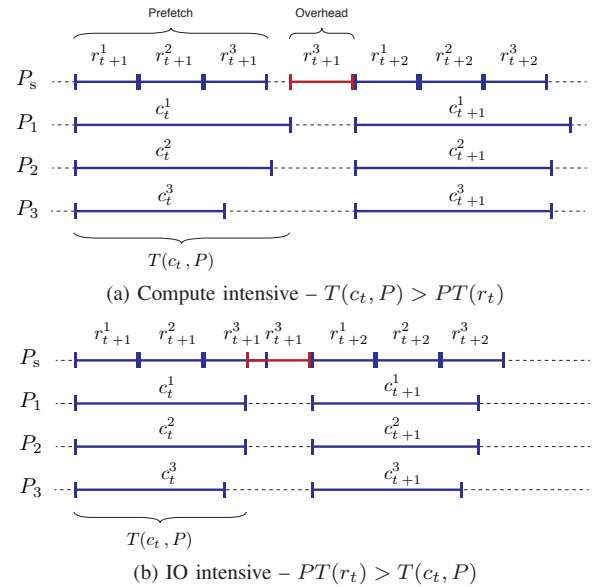


Fig. 7: 1:N ADIOS staging service model where 1 staging process (P_s) handles IO requests from P compute processes (P_n). Here we depicts N=3 case. With pre-fecting in P_s , we can overlap computing time and IO time.

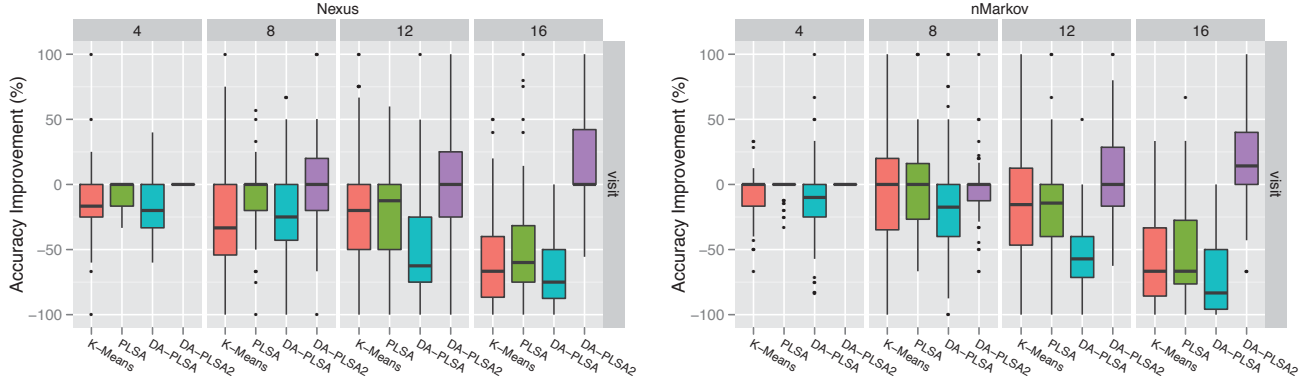


Fig. 6: Impacts of context mining algorithms on the prediction performance of the Nexus algorithm, nMarkov, for the *VisIt* dataset. The DA-PLSA algorithm with 2 mode membership (DA-PLSA2) shows the best performances.

can define the base time T_{Base} , total execution time without prefetching, can be defined as a sum of sequential reading and parallel computing time, such that,

$$T_{Base} = P \sum_n T(r_t) + \sum_n T(c_t, P) \quad (9)$$

Hereafter, for brevity, we ignore superscripts indicating the process indices and assume an uniform reading and computing time through steps.

Now we consider two types of workload: A) compute intensive case when $T(c_t, P) \geq PT(r_t)$ (Fig. 7a), and B) IO intensive case, otherwise (Fig. 7b).

Compute intensive case (Case A): P_s has enough time for pre-fetching N data sets for the $t + 1$ step while N P_n 's running concurrently in t -th step. After P_n 's finishing computing, incorrectly pre-fetched data need to be re-fetched, which occurs overheads. Let denote α be the average accuracy of pre-fetching in P_s . Compared with T_{Base} , we can save $\alpha N \sum_t T(r_t)$ time due to the pre-fetching. The expected execution time with pre-fetching, $T_{Prefetch-A}$, is defined by

$$T_{Prefetch-A} = T_{Base} - \alpha N \sum_t T(r_t) \quad (10)$$

IO intensive case (Case B): P_s has no enough time for pre-fetching N data sets during the computing time $T(c_t, P)$. Re-fetching for incorrect prefetched data can be occurred immediately after $T(c_t, P)$. Compared with the base time T_{Base} , we can save on average $\alpha T(c_t, P)$ with the pre-fetching accuracy α . We can define the expected execution time by

$$T_{Prefetch-B} = T_{Base} - \alpha \sum_t T(c_t, P) \quad (11)$$

We denote γ be the ratio of workload; either $\gamma = T(c_t, P)/NT(r_t)$ (Case A) or $\gamma = NT(r_t)/T(c_t, P)$ (Case B), so that $\gamma \geq 1$. Then, we can define speedup $S_{prefetch}$, time improvement ratio due to the prefetching, by

$$S_{prefetch} = \frac{T_{Base}}{T_{Prefetch}} = \frac{1 + \gamma/N}{1 + \gamma/N - \alpha} \quad (12)$$

for both cases.

With Eq. (12), we can estimate the maximum speedup we can achieve with different pre-fetching accuracy levels.

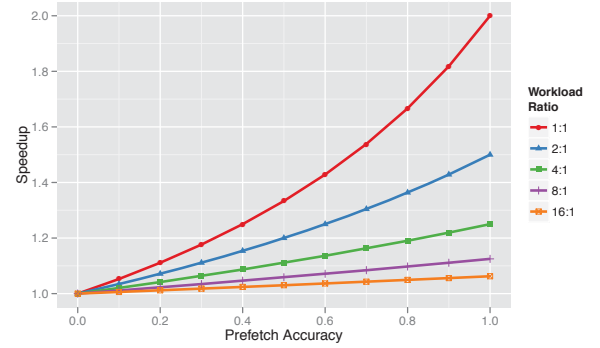


Fig. 8: Estimated speedup improvement by using DA-PLSA with respect to the different level of prefetch accuracy.

In Fig. 8, a simulation result is summarized with various workload ratios ($\gamma = 1, 2, \dots, 16$) and prefetching accuracies ($\alpha = 0.0, \dots, 1.0$). What Fig. 8 implies is as follows:

- Ideally, we can achieve maximum 2 times speedup when the prefetch accuracy is 1.0 with an uniform compute-IO workload (i.e., $\gamma = 1.0$), since we can overlap IO and computing time without any overhead.
- If the workload get skewed ($\gamma > 1.0$), the speedup will be degraded, which is expected.
- High prefetching accuracy contributes more to large speedup, while low prefetching accuracy does less to speedups; Even small increase of prefetching accuracy is important in a balanced workload execution to achieve large speedup.

VI. CONCLUSION

Today, I/O has become a significant source of performance bottleneck for scientific applications. Deploying a predictive data pre-fetcher has been considered as a viable solution to load data in before real requests happen. Especially in scientific data visualization, pre-fetching can be used to reduce the IO waiting time.

To support predictive pre-fetching to reduce IO latency, in this paper we have presented two main solutions. We have developed and demonstrated a provenance system built on ADIOS, named ADIOS-P, that can collect file and variable access patterns for data mining.

We have also proposed a data mining technique for discovering the hidden contexts in data access patterns to improve the prediction performance. More specifically, we applied Probabilistic Latent Semantic Analysis (PLSA), a mixture model based algorithm popular in the text mining area, to mine hidden contexts from the collected user access patterns and, then, we run a predictor within the discovered context. We further improved the PLSA algorithm by applying the Deterministic Annealing (DA) method to overcome the local optimum problem from which the original PLSA algorithm suffered.

We also demonstrated performance results of our DA-optimized PLSA, named DA-PLSA, compared with the original EM-optimized PLSA and presented experimental results showing improvements in prefetching accuracy by using two data sets; DFSTrace file access traces and variable access logs collected from the visualization software, VisIt, through ADIOS-P.

REFERENCES

- [1] Top500.org. (2012, July) Top500 list - november 2000 (1-100). [Online]. Available: <http://www.top500.org/list/2000/11/>
- [2] ——. (2012, July) Top500 list - june 2012 (1-100). [Online]. Available: <http://www.top500.org/list/2012/06/100>
- [3] J. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, and C. Jin, "Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS)," in *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*. ACM, 2008, pp. 15–24.
- [4] J. H. Chen *et al.*, "Terascale direct numerical simulations of turbulent combustion using S3D," *Comp. Sci. & Disc.*, vol. 2, no. 1, p. 015001 (31pp), 2009.
- [5] M. Rienecker, M. Suarez, R. Todling, J. Bacmeister, L. Takacs, H. Liu, W. Gu, M. Sienkiewicz, R. Koster, R. Gelaro *et al.*, "The geos-5 data assimilation system: Documentation of versions 5.0. 1, 5.1. 0, and 5.2. 0," *NASA Tech. Memo*, vol. 104606, 2007.
- [6] S. Klasky, S. Ethier, Z. Lin, K. Martins, D. McCune, and R. Samtaney, "Grid-based parallel data streaming implemented for the gyrokinetic toroidal code," in *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, 2003, p. 24.
- [7] C. S. Chang, S. Klasky *et al.*, "Toward a first-principles integrated simulation of Tokamak edge plasmas - art. no. 012042," *Scidac 2008: Scientific Discovery through Advanced Computing*, vol. 125, pp. 12042–12042, 2008.
- [8] H. Childs, D. Pugmire, S. Ahern, B. Whitlock, M. Howison, G. Weber, E. Bethel *et al.*, "Extreme scaling of production visualization software on diverse architectures," *Computer Graphics and Applications, IEEE*, vol. 30, no. 3, pp. 22–31, 2010.
- [9] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1999, pp. 50–57.
- [10] —, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, vol. 42, no. 1, pp. 177–196, 2001.
- [11] J. Choi, "Unsupervised learning of finite mixture models with deterministic annealing for large-scale data analysis," Ph.D. dissertation, Indiana University, 2012.
- [12] N. Ueda and R. Nakano, "Deterministic annealing em algorithm," *Neural Networks*, vol. 11, no. 2, pp. 271–282, 1998.
- [13] L. Mummert and M. Satyanarayanan, "Long term distributed file reference tracing: Implementation and experience," *Software: Practice and Experience*, vol. 26, no. 6, pp. 705–736, 1996.
- [14] P. Gu, Y. Zhu, H. Jiang, and J. Wang, "Nexus: a novel weighted-graph-based prefetching algorithm for metadata servers in petabyte-scale storage systems," in *Cluster Computing and the Grid, 2006.*, vol. 1. IEEE, 2006, pp. 8–pp.
- [15] P. Gu, J. Wang, Y. Zhu, H. Jiang, and P. Shang, "A novel weighted-graph-based grouping algorithm for metadata prefetching," *Computers, IEEE Transactions on*, vol. 59, no. 1, pp. 1–15, 2010.
- [16] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.
- [17] G. Furnas, S. Deerwester, S. Dumais, T. Landauer, R. Harshman, L. Streeter, and K. Lochbaum, "Information retrieval using a singular value decomposition model of latent semantic structure," in *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1988, pp. 465–480.
- [18] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.
- [19] T. Kroeger and D. Long, "The case for efficient file access pattern modeling," in *Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on*. IEEE, 1999, pp. 14–19.
- [20] —, "Design and implementation of a predictive file prefetching algorithm," in *Proceedings of the 2001 USENIX Annual Technical Conference*, 2001, pp. 105–118.
- [21] I. Choi and C. Park, "Enhancing prediction accuracy in pcm-based file prefetch by constrained pattern replacement algorithm," *Computational Science—ICCS 2003*, pp. 714–714, 2003.
- [22] B. Gill and L. Bathen, "Amp: adaptive multi-stream prefetching in a shared cache," in *Proceedings of the 5th USENIX conference on File and Storage Technologies*. USENIX Association, 2007, pp. 26–26.
- [23] M. Li, E. Varki, S. Bhatia, and A. Merchant, "Tap: Table-based prefetching for storage caches," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies*. USENIX Association, 2008, p. 6.
- [24] X. Fang, O. Sheng, W. Gao, and B. Iyer, "A data-mining-based prefetching approach to caching for network storage systems," *INFORMS Journal on computing*, vol. 18, no. 2, pp. 267–282, 2006.
- [25] D. Joseph and D. Grunwald, "Prefetching using markov predictors," *ACM SIGARCH Computer Architecture News*, vol. 25, no. 2, pp. 252–263, 1997.
- [26] K. Nesbit and J. Smith, "Data cache prefetching using a global history buffer," in *High Performance Computer Architecture, 2004. HPCA-10. Proceedings. 10th International Symposium on*. Ieee, 2004, pp. 96–96.
- [27] G. Pallis, A. Vakali, and J. Pokorny, "A clustering-based prefetching scheme on a web cache environment," *Computers & Electrical Engineering*, vol. 34, no. 4, pp. 309–323, 2008.
- [28] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2210–2239, 1998.
- [29] K. Rose, E. Gurewitz, and G. Fox, "A deterministic annealing approach to clustering," *Pattern Recognition Letters*, vol. 11, no. 9, pp. 589–594, 1990.
- [30] T. Hofmann and J. Buhmann, "Pairwise data clustering by deterministic annealing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 1, pp. 1–14, 1997.
- [31] X. Yang, Q. Song, and Y. Wu, "A robust deterministic annealing algorithm for data clustering," *Data & Knowledge Engineering*, vol. 62, no. 1, pp. 84–100, 2007.
- [32] H. Klock and J. Buhmann, "Multidimensional scaling by deterministic annealing," *Lecture Notes in Computer Science*, vol. 1223, pp. 245–260, 1997.
- [33] J. Choi, J. Qiu, M. Pierce, and G. Fox, "Generative topographic mapping by deterministic annealing," *Procedia Computer Science*, vol. 1, no. 1, pp. 47–56, 2010.
- [34] L. Chen, T. Zhou, and Y. Tang, "Protein structure alignment by deterministic annealing," *Bioinformatics*, vol. 21, no. 1, pp. 51–62, 2005.
- [35] G. Fox, "Deterministic annealing and robust scalable data mining for the data deluge," in *Proceedings of the 2nd international workshop on Petascale data analytics: challenges and opportunities*. ACM, 2011, pp. 39–40.
- [36] E. Jaynes, "On the rationale of maximum-entropy methods," *Proceedings of the IEEE*, vol. 70, no. 9, pp. 939–952, 1982.
- [37] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [38] M. Satyanarayanan, J. Kistler, P. Kumar, M. Okasaki, E. Siegel, and D. Steere, "Coda: A highly available file system for a distributed workstation environment," *Computers, IEEE Transactions on*, vol. 39, no. 4, pp. 447–459, 1990.
- [39] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox, "Twister: a runtime for iterative mapreduce," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 810–818.