Copula-HDP-HMM: Non-parametric Modeling of Temporal Multivariate Data for I/O Efficient Bulk Cache Preloading

Lavanya Sita Tekumalla* Chiranjib Bhattacharyya†

Abstract

Caching is an important determinant of storage system performance. *Bulk cache preloading* is the process of preloading large batches of relevant data into cache, minutes or hours in advance of actual requests by the application. We address bulk preloading by analyzing high-level spatio-temporal motifs from raw and noisy I/O traces by aggregating the trace into a temporal sequence of *correlated count vectors*. Such temporal multivariate data from trace aggregation arise from a diverse set of workloads leading to diverse data distributions with complex spatio-temporal dependencies.

Motivated by this, we propose the *Copula-HDP-HMM*, a new Bayesian non-parametric modeling technique based on Gaussian Copula, suitable for temporal multivariate data with arbitrary marginals, avoiding limiting assumptions on the marginal distributions. We are not aware of prior work on copula based extensions of Bayesian non-parametric modeling algorithms for discrete data. Inference with copulas is hard when data is not continuous. We propose a semi-parametric inference technique based on extended rank likelihood that circumvents specifying marginals, making our inference suitable for count data and even data with a combination of discrete and continuous marginals, enabling the use of Bayesian non-parametric modeling, for several data types, without assumptions on marginals.

Finally, we propose *HULK*[1], a strategy for I/O efficient bulk cache preloading using our Copula-HDP-HMM model to leverage high-level spatio-temporal motifs in Block I/O traces. In experiments on benchmark traces, we show near perfect hitrate of 0.95 using HULK, a tremendous improvement over baseline using Multivariate Poisson, with only a fourth of I/O overhead.

1 Introduction

Cache preloading is the process of predicting data to preload into cache before it is requested by the application, to reduce latency in accessing data. *Bulk Cache Preloading* is the process of preloading mega bytes and giga bytes of data at a time, seconds, minutes or hours in advance, before actual accesses are requested.

State-of-the art cache preloading strategies [1, 2], in enterprise storage servers, are predominantly based on capturing locality of space and time based on short range correlations in memory accesses, over extremely short intervals of time (order of milli seconds). By operating in the time scale of minutes and hours unlike traditional caching algorithms, bulk preloading opens up avenues to explore data mining techniques for prediction, that are not as stringently bound by milli-second level timing constraints that are typical of traditional caching techniques. However, there is no work on efficient strategies for bulk cache preloading. The only work in this area [3] fails to make precise predictions, leading to huge I/O overheads, rendering it impractical for real world use cases.

Memory accesses often exhibit long range motifs that span several minutes or even hours (see figure 1), comprising of millions of accesses. For instance, whenever a periodic cron job is run or a movie is played, the same access pattern is likely to repeat over and over albeit with some differences. There is very little work in this area to exploit such long range repeating motifs. Motivated by this, we explore the problem of mining block I/O traces for I/O efficient bulk prediction by leveraging *long-range repeating spatio-temporal motifs* from memory access traces.

Our approach involves spatio-temporal aggregation of raw memory access data into a *sequence of count vectors* to get a higher-level view of the trace. Crucial to our strategy is an adaptive model to capture the latent spatio-temporal correlations in such aggregated count vector sequences, coming from diverse workloads with diverse data distributions. We propose **Copula-HDP-HMM**, to approach the problem of predictive modeling such multivariate data with complex spatio-temporal correlations. Our technique is a Bayesian non-parametric model for temporal multivariate data with arbitrary marginals, that is not limited by assumptions

*lavanya.iisc@gmail.com, Indian Institute Of Science
†chiru@csa.iisc.ernet.in, Indian Institute of Science
[1] cacHing using copULa for bulK preloading

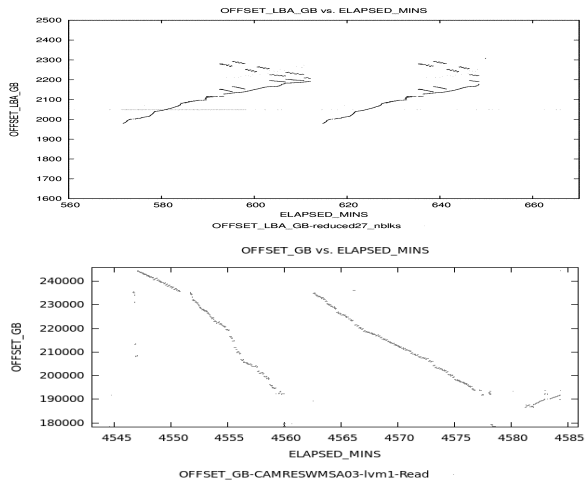


Figure 1: A Visualization of trace (a sequence of access requests), with time on the X axis and the Memory Location accessed on Y axis : This picture shows long range repeating access patterns in traces. In the first picture, we see a long range motif spanning about an hour. In the second picture, we see a motif spanning about 20 minutes.

on marginal distribution.

Bayesian Non-parametric modeling(BNP) is a paradigm that automatically adapts the model complexity depending on the data, a practical requirement to handle a diverse set of traces from different workloads. However, while BNP techniques are well explored for real valued data, and multinomial mixtures, modeling data types such as multivariate counts remains less explored. The only work we are aware of that addresses this problem, models count data with Multivariate Poisson [4, 3], that is unsuitable for higher dimensions [5] owing to its computational intractability and its single parameter limitation, making it unsuitable for overdispersed count data (verified experimentally in sec 6).

More importantly, the current inference methodologies tailored for specific distributions like multivariate Poisson are non-trivial to extend to other data types, or even other multivariate distributions for count data. Motivated by this, the Copula-HDP-HMM extends the HDP-HMM with Gaussian Copula emissions for temporal multivariate data with arbitrary marginals.

Copulas decouple the modeling of correlation structure from choosing the marginal distributions when modeling multivariate data. For instance, to model count data, they allow for a suitable discrete marginal, while the correlation structure is modeled by a suitable copula function, like the Gaussian distribution.

However, existing copula based techniques predominantly work with continuous marginals [6, 7, 8]. Inference for copula based models for count data is a hard problem owing to ties arising when dealing with discrete marginals [9, 10]. Inference attempts in [10] have resulted in algorithms with complexity exponential in the number of dimensions. We propose an inference algorithm based on extended rank likelihood [11], that circumvents specifying the marginals thus rendering our technique robust to the choice of marginals. Hence, our technique is free from limitations that arise due to choosing a specific family of marginals, and learning the parameters of these marginals during the process of inference. For instance, our algorithm is free of the shortcomings of Poisson marginals that lead to problems with over-dispersed data. Apart from this the technique also leads to a general simplification of inference process leading to an efficient inference algorithm that seamlessly works for a variety of data types. We note that our inference algorithm is also suitable for mixed data with a combination of discrete and continuous marginals. Hence, from a caching perspective our copula based approach seamlessly works for other forms of data aggregation, instead of multivariate counts.

Our Contributions:

1. We propose Copula-HDP-HMM by extending the HDP-HMM with Gaussian copula emissions for adaptive modeling of a temporal multivariate data, such as multivariate counts obtained from trace aggregation.

2. We propose an inference algorithm for Copula-HDP-HMM suitable for multivariate data with arbitrary marginals, enabling Bayesian non-parametric modeling for a variety of datatypes, including mixed data with a combination of binary, discrete and continuous marginals. We are not aware of prior BNP models suited for arbitrary marginals more so for temporal data. Inference for copula based models is hard for non-continuous marginals, for instance discrete data such as counts. Our technique circumvents specifying marginals making it suitable for a variety of datatypes. Our technique addresses a fundamental machine learning problem, of adaptive temporal mixture models for multivariate data with arbitrary marginals with wide applicability outside the caching domain.

3. We propose HULK : *Caching Using Copulas for I/O efficient Bulk Preloading*, leveraging repeating long-range motifs in traces with the Copula-HDP-HMM to make precise predictions with minimal I/O overhead. HULK works well when traces exhibit long-range spatio temporal motifs, such as demonstrated in Figure 1. The proposed non-parametric Copula based methodologies naturally adapt to such spatio-temporally correlated patterns giving substantially high speed-ups as demonstrated in experiments on benchmark traces.

Organization: Our model Copula-HDP-HMM and requisite background is described in section 2,3,4. In sec 5 we propose HULK, our novel Bulk cache preloading

framework using the Copula-HDP-HMM.

2 Preliminaries

We now describe some background on Copulas.

Copula: A copula enables modeling a multivariate joint distribution as a function of the individual marginal distributions of variables. It enables separation of modeling the dependence structure of the data from modeling the marginals. In other words given M-dimensional multivariate data, the M marginals can be chosen based on those most suitable for the application, while their dependence structure can be independently modeled by the underlying copula function. Consider a dataset $\{X_t\}_{t=1}^{T}$ with T instances of M dimensional random vectors. (For trace modeling, each X_t is a count vector: $X_t \in \mathbb{Z}^M$ obtained as a result of aggregation.) Let $U_{tj} = F_j(X_{tj})$ be the random variable corresponding to the marginal distribution $F_j(.), \forall j$, while u_{tj} (small case) denote a specific value taken by $U_{tj}, \forall t, j$.

Formally, a copula for random vector $(X_{t1}, \ldots, X_{tM})$ is defined as the joint cumulative distribution of random variables $U_{t1} = F_1(X_{t1}), \ldots, U_{tM} = F_M(X_{tM})$ as follows:

$$C(u_{t1}, \ldots, u_{tM}) = P(U_{t1} \leq u_{t1}, \ldots, U_{tM} \leq u_{tM})$$

Gaussian Copula enables modeling the correlation structure with Gaussian distribution. Given observed random vector $\mathbf{X} = (X_{t1}, \ldots, X_{tM})$, let $Y_{t1} = \phi_{std}^{-1}(F_1(X_{t1})), \ldots, Y_{tM} = \phi_{std}^{-1}(F_M(X_{tM}))$ where ϕ_{std} is the standard normal distribution, the Gaussian copula is defined generatively:

$$Y_{t1}, \ldots, Y_{tM} \sim \mathcal{N}(0, \Sigma) \text{where}$$
$$X_{tj} = F_j^{-1}(\phi_{std}(Y_{tj})), j \in [M] \tag{2.1}$$

The theoretical foundation for the application of copulas is provided by Sklars theorem [12]: Given the joint cumulative distribution of a random vector with continuous marginals, there exists a unique copula corresponding to it. [13] discusses several parameter estimation techniques for copula with continuous marginals. However, inference for discrete marginals is hard due to ties as described in[9].

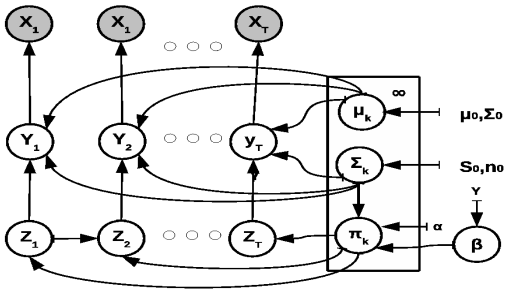
2.1 Related Work for Copula-HDP-HMM:

We now briefly review prior work related to Copula-HDP-HMM.There has been no prior work on exploring Bayesian non-parametric modeling algorithms for multivariate data with arbitrary marginals. Specifically, for multivariate count data, that arises from our caching application, the only prior technique for non-parametric temporal mixture modeling, that we are aware of, is based on the Multivariate Poisson(MVP)[4, 3]. However, the MVP, and even sparse-MVP[3] lead to quadratic latent variables per data point, leading to overfitting and a huge computational overhead during inference (Copula based approach on the contrary is linear with $O(M)$ latent variables). More importantly, the Poisson is also limited in its modeling power due to the single parameter nature, making it unsuitable for overdispersed data, a significant shortcoming for count data obtained from aggregation of traces. Further, inference strategies tailored for Poisson cannot easily be extended to more suitable alternatives such as the negative-binomial[14]. Copula based techniques have been widely explored for modeling multivariate data [13], but predominantly for data with continuous marginals [7, 8]. Dealing with arbitrary marginals, such as data with discrete marginals is harder due to ties [9, 10]. Inference attempts for discrete data have resulted in exponential complexity[10].

Our inference procedure bypasses the specification of marginal leading to a simplification of the inference process and makes our techniques suitable for multivariate data with arbitrary marginals including a combination of discrete and continuous marginals. We now present our proposed model Copula-HDP-HMM in section 3, while we discuss inference in section 4.

3 Copula-HDP-HMM : Non-parametric Model for Temporal Multivariate Data

Non-parametric temporal mixture models for multivariate data is an important problem with wide applicability. We propose a new technique, the Copula-HDP-HMM, a temporal Dirichlet Process mixture model for multivariate data with Gaussian Copula. Inference is not straightforward[9] for non-continuous marginals. We are not aware of prior work that addresses copula based Bayesian non-parametric modeling techniques for multivariate data with arbitrary marginals. The generative model for Copula-HDP-HMM is shown in Alg 1.



The Copula-HDP-HMM model, through a HDP prior for the transition matrix of the HMM, is non-parametric in the number of states. The $\pi_k, \forall k = 1, \ldots$ are the transition probabilities from state k to other

Algorithm 1: Copula-HDP-HMM Generative Model for adaptive clustering of temporal multivariate data

$\beta \sim GEM(\gamma)$
for $k = 1, \ldots$ **do**
  $\pi_{\mathbf{k}}|\beta, \alpha \sim DP(\alpha, \beta)$
  $\mu_{\mathbf{k}} \sim \mathcal{N}(\mu_{\mathbf{0}}, \mathbf{\Sigma_0})$
  $\Sigma_k \sim InvWish(S_0, n_0)$
for $t = 1, \ldots, T$ **do**
  $Z_t|Z_t - 1, \pi \sim \pi_{Z_{t-1}}$
  $\mathbf{Y}_t|Z_t, \mu, \mathbf{\Sigma} \sim \mathcal{N}(\mu_{\mathbf{k}}, \mathbf{\Sigma_k})$
  for $j = 1, \ldots, M$ **do**
    $X_{tj} \sim F_j^{-1}(Y_{tj})$

states. They come from a HDP[15] with base distribution β and a concentration parameter α to allow for transition to a countably infinite possible states. Note that β itself comes from a stick breaking construction with a concentration parameter γ. $Z_t, \forall t$ in our algorithm denote the mixture component(cluster) assignment indices or the index of a hidden state for a particular data point t. For each cluster k the set of HMM emissions $\{\mathbf{Y}_t : Z_t = k\}$ come from a Gaussian Copula with a mean μ_k and variance Σ_k that have a multivariate Gaussian and an Inverse Wishart prior respectively. The choice of these priors arises from conjugacy, simplifying the inference process. We note that while correlation structure is captured by the Gaussian, the actual data $X_t, \forall t$ is obtained from the quantile function (by inversion) of the corresponding marginals $F_j(), \forall j$. However, we do not explicitly estimate these marginals by adopting the rank likelihood approach, making our clustering robust to the choice of marginals.

We note that we differ from the standard definition of the zero mean Gaussian Copula in literature, since in our clustering usecase, we would like to ensure a separation between means of Gaussians to better cluster the data. Hence we explicitly model the means in a Gaussian Copula and even place a prior for the means.

4 Inference with Extended Rank Likelihood

While inference for HDP-HMM is well explored, Copula-HDP-HMM leads to interesting challenges due to Copula based emissions with arbitrary non-continuous marginals.We now outline an inference algorithm for Copula-HDP-HMM with Gibbs sampling that is applicable for discrete as well as mixed data.

To avoid the expensive step of marginal estimation, we use extended rank likelihood [11] that enables inferring copula parameters based on the rank information of data, bypassing specifying marginals. Consider the dataset $\{X_t\}_{t=1}^T$ with T instances of M dimensional random vectors generated from a multivariate distribution. For the case of a single Gaussian Copula distribution (eqn 2.1), without any knowledge of the marginals $F_j, \forall j$ and without observing $\mathbf{Y}$, observing the set of instances $\mathbf{X}$ tells us that each $Y_t, \forall t$ must lie in the set that preserves rank order: $\mathbf{D} = \{\mathbf{Y} : \max\{Y_{sj} : X_{sj} < X_{tj}\} < X_{tj} < \min\{Y_{sj} : X_{tj} < X_{sj}\}, \forall t, j\}$ The occurrence of this event is considered to be our data in this inference technique.

We now propose Gibbs Sampling inference for the Copula-HDP-HMM model discussed in algorithm 1, that involves the use of extended rank likelihood in a Dirichlet Process mixture model setting for the first time where we have additional rank constraints arising from the clustering. We focus mainly on the updates for $\mathbf{Y}$ and $\mathbf{Z}$ in algorithm 1 in detail, that highlight the new challenges due to the use of extended rank likelihood in a mixture model setting. Sampling of the random variables μ and $\mathbf{\Sigma}$ follow from the conjugacy of their chosen priors, while the update for β, is similar to that described in [15, 16] through the use of auxiliary variables $\mathbf{m}$ introduced for HDP inference.

We introduce some notation for the rest of this section. $\mathbf{Y} = \{\mathbf{Y}_t : t \in [T]\}$, $\mathbf{Y}^k = \{\mathbf{Y}_t : t \in [T], Z_t = k\}$, $\mathbf{Z} = \{\mathbf{Z}_t : t \in [T]$ and $\mathbf{X} = \{\mathbf{X}_t : t \in [T]$. A set with a subscript starting with a hyphen$(-)$ indicates the set of all elements except the index following the hyphen. Finally, let $n_k = Cardinality\{\mathbf{X}_t : Z_t = k\}$ We also introduce auxiliary variables $m_{.k}$, that arise from inference for the underlying HDP (denoting the cardinality of the partitions generated by the base DP). These auxiliary latent variables aid the sampling of β based on the direct sampling procedure from HDP[15].

Sampling Y : The update of the $\mathbf{Y}_t$ entails conditioning on the rank constraints instead of the observed data due to the Rank likelihood based inference approach. Further, in a mixture modeling setting, given $Z_t = k$, we would like to constrain $\mathbf{Y}_t$ such that elements of $\mathbf{Y}^k$ after the update adhere to the original rank order of observations in its cluster.

Formally, let $\mathbf{D}_k$ be the set of all $\mathbf{Y}^k$ that preserve the rank order of the observed data for cluster k. Hence, $\mathbf{D}_k = \{\mathbf{Y}_t \in \mathbf{Y}^k : \max\{Y_{rj} : Z_r = k, X_{rj} < X_{tj}\} < X_{tj} < \min\{Y_{rj} : Z_r = k, X_{tj} < X_{rj}\}, \forall j\}$. We would like to sample $\mathbf{Y}_t \sim \mathcal{N}(\mu_k, \Sigma_k)|\mathbf{Y}_t \in \mathbf{D}_k$. Hence, the Gibbs sampling update for $\mathbf{Y}_t$ is based on sampling from a multivariate Gaussian distribution that is truncated to maintain the rank constraints within cluster k. Given $Z_t = k$, the lower and upper limit are computed as

$$\text{Let } (Y_{tj}^L)_k = max\{Y_{rj} : Z_r = k, X_{rj} < X_{tj}\}$$
$$(Y_{tj}^U))_k = min\{Y_{rj} : Z_r = k, X_{tj} < X_{rj}\}$$

Based on these limits, the update for $\mathbf{Y}_t$ is computed as

$$\mathbf{Y}_t \sim \mathcal{N}(\mu_k, \Sigma_k)|(Y_{tj}^L)_k < Y_{tj} < (Y_{tj}^U)_k, \forall j \in [M] \quad (4.2)$$

Sampling Z : For Sampling Z, the cluster assignment differs from the standard approach [17] due to the Rank constraint underlying the inference algorithm. We note from the previous update of $\mathbf{Y}_t$ that while we ensure the rank constraints for $\mathbf{Y}^k$ are satisfied in each cluster, we allow leeway so that they need not be satisfied across clusters. Hence, for any $k \in [K]$, Z_t being set to k is permissible if $\mathbf{Y}^k \cup \mathbf{Y}_t$ meets the rank constraints, i.e let the set of permissible clusters be $\Delta_t = \{k : (Y^L_{tj})_k < Y_{tj} < (Y^U_{tj})_k, \forall j \in [M]\}$. The update for Z_t is computed similar to the update for the HDP-HMM (refer to [17] for details), however the support for the update is constrained to the set Δ_t for selecting an existing cluster.

$$
\begin{aligned}
&p(Z_t = k | Z_{-t,-(t+1)}, z_{t+1} = l, \beta, Y) \\
&\propto p(z_t = k | z_{-t,(t+1)}; \alpha) p(z_{t+1} = l | z_t = k, z_{-t,(t+1)}; \alpha) \\
&\qquad p(\mathbf{Y}_t | Z_t = k) \delta(k \in \Delta_t) \qquad (4.3)
\end{aligned}
$$

Computing the probability of Z_t taking a new component k_{new} requires integrating over the prior distributions of μ_{new} and Σ_{new}. We follow the technique proposed by Neal [18], used by [19] in a similar setting by finding a Monte Carlo estimate of the probability of selecting a new cluster.

Algorithm 2: Inference: Copula-HDPHMM

repeat
  for $t = 1, \ldots, T$ **do**
    Sample Z_t from Eqn 4.3
    for $j \in [M]$ **do**
      Sample Y_{tj} from Eqn 4.2
  for $k = 1 \ldots, K$ **do**
    Sample $\mu_k \sim p(\mu_k | \mathbf{Y}^k; \mu_0, \Sigma_0)$ (see suppl)
    Sample $\Sigma_k \sim p(\Sigma_k | \mathbf{Y}^k, \mu_k; S0, n0)$ (see suppl)
  for $k = 1 \ldots, K$ **do**
    $m_k = 0$
    for $i = 1, \ldots, n_k$ **do**
      $u \sim Ber(\frac{\alpha\beta_k}{i+\alpha\beta_k})$, if $(u == 1) m_k++$
  $[\beta_1 \beta_2 \ldots \beta_K \beta_{K+1}] | m, \gamma \sim Dir(m_1, \ldots, m_k, \gamma)$
until *convergence* ;

Sampling μ and Σ: Updates for mean μ_k and variance Σ_k, $k \in [K]$, can be evaluated in closed form due to conjugacy of priors. (more in supplementary).

5 HULK : A strategy for I/O Efficient Bulk Cache Preloading

We now propose HULK (cacHing with copUlas for buLK preloading), a new framework for bulk cache preloading leveraging long range motifs in I/O traces. On a storage server, we define *live trace* as a live set of memory access requests being received, while, *repository trace* is a set of requests collected apriori.

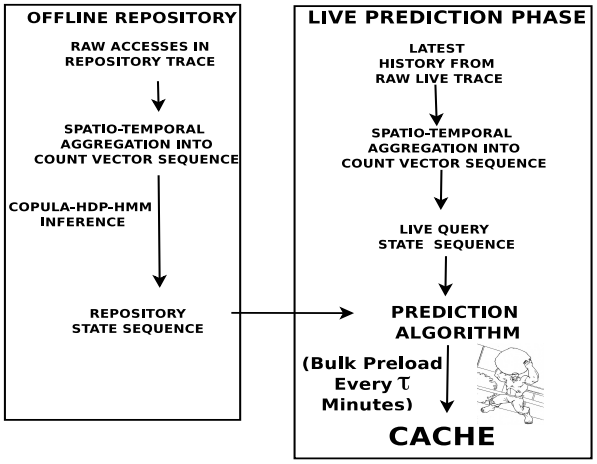


Figure 2: *HULK : Our Bulk Cache Preloading Strategy*

Overview of HULK : HULK learns by analyzing the repository trace and predicts future accesses in live trace by observing a brief history of live trace and comparing with prior accesses in repository. Hence, HULK has two aspects (1) Offline repository processing by learning with Copula-HDP-HMM (2)Bulk prediction on live trace

–*The offline repository processing* involves spatio temporal aggregation of repository trace into a sequence of count vectors followed by Copula-HDP-HMM inference that further reduces the count vectors into a concise sequence of latent state IDs to be used during prediction. (This phase runs offline. Hence no time constraints)

–*Bulk Prediction on live trace* involves examining a fixed history of live trace to identify if it is a repetition of prior pattern from repository and leveraging it to make bulk predictions periodically. For efficient processing, the live trace history is also spatio-temporally aggregated into a sequence of count vectors and subsequently condensed to a sequence of latent state IDs on which prediction algorithm runs. (In our experiments, bulk preload happens every half minute, while prediction algorithm takes less than a second to determine preload candidates.)

Our workflow is shown in fig 2. The count modeling technique during repository phase is a crucial determinant of effectiveness of HULK in terms of its predictive ability. To make modeling effective across any server generating any IO trace, we need a technique without limiting assumptions about marginals of count vectors, at the same time adaptive in the number of latent states discovered based on amount of detail in the I/O trace and based on underlying workload and nature of correlations motivating the design of Copula-HDP-HMM.

In section 5.1 we first review relevant work on bulk cache-preloading. In section 5.2 and section 5.3, we describe various aspects of HULK in more detail.

5.1 Related Work: Bulk Cache Preloading There is very little prior work [20, 3] for bulk preloading by analyzing block I/O traces. Bonfire [20], explores a very specific form of bulk preloading to warm a cold cache. We address a different and a more general problem for an active cache. In [3], the authors attempt trace analysis by leveraging temporal correlations with a multivariate Poisson with a rudimentary HMM based prediction strategy that leads to excessive I/O overhead. HULK significantly differs from this approach, through its Copula-HDP-HMM based technique that shows significant gains and employs sequence alignment strategies to explicitly find pattern repetitions leading to I/O efficient predictions unlike [3]. There has been work that addresses preloading for extremely short time spans in the range of microseconds [1, 21, 2]. In [1], the authors specialize in workloads with notable sequential accesses, while in [2], the authors further use application level contexts for dealing with interleaved workloads. Our work differs, in that we look at general non-sequential workloads, without assuming any application level context. Moreover, such algorithms[1, 21], that use short range correlations, can typically be used alongside bulk cache preloading strategies for more performance gains.

5.2 Spatio-Temporal Aggregation: From Raw Trace to Temporal Sequence of Count Vectors Temporal aggregation (a.k.a abstraction) [22], is a technique [23, 24] to infer high-level concepts from raw temporal data. We perform spatio-temporal aggregation, aggregating the trace both temporally and spatially.

Temporal Aggregation: A trace is a sequence of raw requests, $r_q =< u_q, b_q, n_q >$, $\forall q \in [Q]$, comprising of triplets of the form $< Timestamp, BlockID, \#BlocksRequested >$ for each request[2]. We would like to temporally aggregate the trace into *time-slices* of length τ minutes, to obtain say N slices. Hence, the outcome of the temporal aggregation phase, is a sequence of sets $(A_1, \ldots, A_N)$, such that each set A_t contains the set of requests in the t^{th} timeslice. (In other words $A_t = \{r_q : (t-1)\tau \leq u_q < t\tau, q \in [Q]\}$.)

Spatio-Temporal Aggregation: We now spatially aggregate each slice A_t of the trace by taking a histogram over ν sized bins, M in number. Each slice is now spatially aggregated into histograms (or count vectors) $X_t \in \mathbb{Z}^M$ with M bins such that X_{tj} is the count of all accesses falling into bin j in slice t. (In other words, $X_{tj} = Cardinality\{r_q \in A_t : (j-1)\nu \leq b_q < j\nu\}$). The outcome of this step is a temporal sequence of count vectors $(X_1, \ldots, X_T)$, where $\forall t, X_t \in \mathbb{Z}^M$.

[2] $\forall N \in \mathbb{Z}$, we use $i \in [N]$to denote $i \in \{1, \ldots, N\}$

The sequence of Count vectors thus obtained through spatio-temporal aggregation $(X_1, \ldots, X_T)$ is input to our model, Copula-HDM-HMM, described in 3.

5.3 HULK for Bulk Cache Preloading: HULK comprises of two parts (1) The Offline Repository Processing & (2) Real-Time Prediction for a Live Trace.

5.3.1 Offline Repository Processing Offline repository processing, operating on previously collected trace data, is a learning phase, run in a batch mode (for instance once a day), offline, running for minutes or even hours. First we preprocess past traces by spatio-temporally aggregating data to create a temporal sequence of count vectors $(X_1, \ldots, X_{T_{repo}})$. Next is temporal modeling with the proposed Copula-HDP-HMM (see sec 3) to obtain a concise representation of the trace as a sequence of hidden HMM states $\mathcal{Z}_R = (Z_1, \ldots, Z_{T_{repo}})$ corresponding to the input sequence. Note that T_{repo} is the number of timeslices in the repository with each $Z_t \in \{1, \ldots, K\}$, where K is the number of hidden states recovered by the model.

The outcome of the repository processing step is a sequence of sets of accesses in each time slice $(A_1, \ldots, A_{T_{repo}})$ through spatio-temporal aggregation and a sequence of state IDs, $\mathcal{Z}_R$.

5.3.2 Real-Time Prediction On Live Trace In prediction phase, the history of live trace is analyzed by comparing with the repository to make predictions for preload. Preload predictions are made every τ minutes and our prediction algorithm is designed to run in time *much less* (order milli secs) than τ to ensure timely preload. The preloader loads all predicted accesses in bulk and repeats the process after τ minutes.

As the first step in this process, live trace is subject to spatio-temporal aggregation (sec 5.2) to obtain a sequence of count vectors $(\bar{X}_1, \ldots, \bar{X}_{T_{live}})$. These are further aggregated into a concise sequence of state IDs of the form $\bar{\mathcal{Z}}_L = (\bar{Z}_1, \ldots, \bar{Z}_{T_{live}})$, $\bar{Z}_t \in \{1, \ldots, K\}, \forall t \in [T_{live}]$ using viterbi algorithm with learnt Copula-HDP-HMM model from offline repository step, where T_{live} is length of live trace history. The bar distinguishes variables of live phase from repository phase.

The *prediction algorithm* aligns a sequence of state IDs of live trace with the sequence of state IDs in the repository to identify repeating motifs for preload (runs with comlexity $O(T_{live}T_{repo})$). The input to prediction algorithm comprises of a sequence of state IDs for the live trace history $\bar{\mathcal{Z}}_l$ a sequence of state IDs for the repository trace $\mathcal{Z}_R$. The aim of prediction algorithm is to find the best locally aligned subsequence of $\mathcal{Z}_L$, in the live trace history, with the sequence $\mathcal{Z}_R$ (the repo in

its entirety). If the algorithm finds an alignment point, it returns a set $P \subset [T_{live}]$ of slice indices, representing a window of slices around the point of alignment, that are the best candidates for preload. This is described in the appendix Alg 3. All accesses $\{\bigcup_{\forall t \in P} A_t\}$ corresponding to these repository slices are loaded into cache.

6 Experimental Evaluation

We discuss our experiments to evaluate Copula-HDP-HMM model and HULK framework. Our code and dataset available at *http://bit.ly/HulkSmashHitCache*

6.1 Experiment 1 Data with Various Marginals

We first evaluate the ability of Copula-HDP-HMM to model synthetic data with various marginals by correctly identifying hidden states. The four datasets comprise of data with negative binomial, binomial, two datasets with Poisson marginals, with gold standard labels for hidden states. (Sec D of suppl has details of data generation). Our baselines are (1) IP-HDP-HMM, the simplest conceivable model for non-parametric temporal modeling of count data with independent Poisson emissions across features, (2) SMVP-HDP-HMM[3], the only available non-trivial baseline for HDP-HMM extension with multivariate counts. We see that the Copula-HDP-HMM significantly outperforms these Poisson based baselines. While baselines are comparable for Poisson2 marginals, when data is generated with binomial or the negative-binomial, Copula-HDP-HMM clearly wins.

Table 1: Normalized Mutual Index(NMI): Average values on 25 iterations (Standard Deviation in brackets). Higher the better.

Marginal Type	Copula-HDP-HMM	SMVP-HDP-HMM	IP-HDP-HMM
Binomial	**0.93**(sd:0.007)	0.85(sd:0.004)	0.84(sd:0.001)
Neg-Binomial	**0.91**(sd:0.009)	0.86(sd:0.003)	0.85(sd:0.001)
Poisson1	**0.99**(sd:0.008)	0.92(sd:0.070)	0.85(sd:0.002)
Poisson2	**1.00**(sd:0.000)	1.00(sd:0.000)	0.85(sd:0.040)

Table 2: Adjusted Rand Index(ARI): Average values on 25 iterations (Standard Deviation in brackets). Higher value better.

Marginal Type	Copula-HDP-HMM	SMVP-HDP-HMM	IP-HDP-HMM
Binomial	**0.86**(sd:0.014)	0.72(sd:0.023)	0.70(sd:0.004)
Neg-Binomial	**0.84**(sd:0.016)	0.72(sd:0.012)	0.70(sd:0.002)
Poisson1	**0.99**(sd:0.007)	0.86(sd:0.128)	0.70(sd:0.006)
Poisson2	**1.00**(sd:0.000)	1.00(sd:0.000)	0.69(sd:0.024)

6.2 Evaluation on I/O traces:

We perform experiments to evaluate Copula-HDP-HMM through our strategy HULK in comparison with baseline models on real world and synthetic I/O traces in terms of hitrates and I/O efficiency.

Trace DataSets: We use the dataset of publicly available Microsoft Enterprise Server Storage Traces(MSR Cambridge traces) as that used in [25, 20] and an industrial trace. These traces are selected from a diverse set of workload types, such as SQL Server, Source control, Print Servers and so on. We also generate synthetic traces with different characteristics. (Details on these traces are shown in suppl material).

HULK Baselines: Intelligent algorithms for Bulk Cache Preloading is a field in its infancy. Hence we use baseline that performs no preload and those that use Poisson marginals for preload.

–*Baseline-SMVP-Preload*: We use HDP-HMM based on Sparse-Multivariate-Poisson for trace modeling as baseline, based on the only prior attempt at modeling traces[3] for bulk preload leading to huge I/O overhead

–*Baseline-IP-Preload*: Uses a simpler model, the HDP-HMM with Independent Poisson emissions that models all count vector dimensions independently.

–*∅-Preload*: Null-Preload, involves no bulk preload. Demonstrates effectiveness of Bulk Cache Preloading.

6.2.1 Experiment 2: Cumulative Hitrates:

The cache hitrate is an important (and the most straightforward) measure of caching performance. The cache hit rates are defined as:

$$\text{hitrate} = \frac{\text{hits}}{\text{hits} + \text{misses}}$$

See Table 3 for results from this experiment.

Table 3: Comparing Hitrates with various baselines: We note that HULK, our model based on Copula-HDP-HMM outperforms baselines including those based on multivariate Poisson in terms of hitrates. In MS1, with HULK we get a near perfect hitrate of 0.95

Trace Name	HULK Copula Preload	Baseline SMVP Preload	Baseline IP Preload	Baseline ∅ Preload
MS1	**0.95**	0.70	0.52	0.00
MS2	**0.67**	0.49	0.38	0.01
MS3	**0.49**	0.46	0.46	0.39
MS4	**0.05**	0.05	0.05	0.05
MS5	**0.65**	0.63	0.62	0.64
MS6	**0.51**	0.46	0.21	0.03
IN1	**0.62**	0.59	0.24	0.03
SYN1	**0.99**	0.75	0.70	0.00
SYN2	**0.77**	0.60	0.52	0.57
SYN3	**0.85**	0.71	0.59	0.37
SYN4	**0.99**	0.34	0.30	0.06

The Copula-HDP-HMM based preload model (HULK) shows improved hitrates with reduced I/O overhead over all baselines since the Copula based model better captures the spatio-temporal correlations while being free of limitations arising due to choice of Poisson marginals. Both Baseline-SMVP-Preload (multi-

Table 4: Variability in the number of clusters with Copula-HDP-HMM for different traces: Sorted by number of clusters

Trace	MS2	MS6	SYN2	SYN3	MS4	MS1
#Clusters	15	23	25	33	33	34
Trace	MS5	IN1	SYN1	MS3	SYN4	
# Clusters	55	68	90	101	114	

variate Poisson) and Baseline-IP-Preload (Independent Poisson) are based on Poisson marginals leading to sub-optimal grouping of the underlying count vectors that affects the predictive performance, and consequently the hitrates and the I/O efficiency. We also note that Baseline-SMVP-Preload performs better than Baseline-IP-Preload, since the Independent Poisson model is simplistic in treating count vector dimensions as independent. To experimentally validate the need for an adaptive technique using non-parametric modeling, we also report the number of hidden states found by HULK(with Copula-HDP-HMM) during learning phase for different traces. We note a huge variation in this number, validating our need for an adaptive technique that automatically finds the number of hidden states.

We observe that for some of the traces, preloading has a huge improvement in hitrate. For instance, in MS1, preloading through HULK leads to a 0.95 hitrate (close to 100%) and leads to a 35% improvement in hitrate over all the baselines. We also note that all forms of Preload show better hitrate over ∅-Preload, validating the gains in Bulk preloading. We also note, on some traces, there is less improvement in hitrate compared to baselines. This can be attributed to the fact that not all traces have long range motifs.

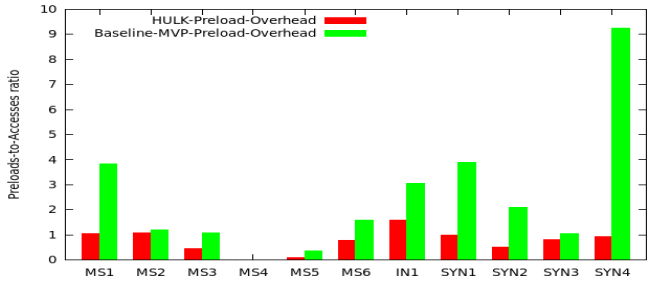


Figure 3: Preloads-To-Accesses ratio: HULK based on Copula-HDP-HMM has a value close to 1 indicating low I/O overhead while baselines have a much higher I/O overhead. On MS1, with HULK, we get only a fourth of I/O overhead compared to baseline

6.2.2 Experiment 3: Preload Overhead: The goal of efficient Bulk cache preloading is to load the right set of blocks at each time slice, that are going to be accessed during the next time slice. We analyze the preload overhead by examining the ratio of number of preloads to the number of accesses in the trace.

$$\text{preloads-to-accesses} = \frac{\text{Number of Preloads}}{\text{Number of accesses}}$$

A high value, well above 1, indicates that a high number of blocks are being preloaded as candidates for each access, indicating a high preload overhead.

We measure the preloads-to-accesses ratio of HULK based on Copula-HDP-HMM in comparison with the best baseline SMVP-Preload from the previous experiment. See Figure 3 for the result of this experiment. We observe that HULK with Copula-HDP-HMM has a preloads-to-accesses ratio close to 1 indicating a low overhead, while our baseline has a very high preloads-to-accesses value. We see that for MS1, the value with our baseline with Sparse-Multivariate-Poisson is close to 4 indicating that the amount of data preloaded is almost 4 times that of the size of the trace unlike our technique. This shows that HULK makes precise predictions and is more I/O efficient than our baseline, consistently, by an order of magnitude.

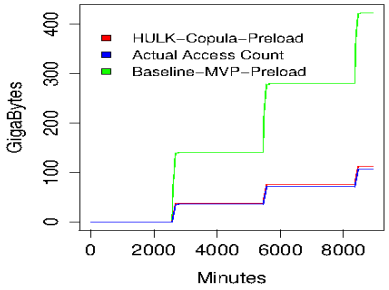


Figure 4: MS1 Case study: Comparison of Cumulative Preloads and actual number of accesses plotted with time. For HULK, we see that number of preloads is very close to the number of accesses indicating that it makes precise preloads with minimal overhead.

6.3 Summary of Results: – In experiment 1 with simulated data with various marginals we see that Copula-HDP-HMM outperforms baselines since it is not restricted by the limitation of choice of marginals.
– In experiment 2, Copula-HDP-HMM based preload model (HULK) shows improved hitrates over all baselines since the Copula based model better captures the spatio-temporal correlations while being free of limitations arising due to choice of Poisson marginals.
– In experiment 3, HULK based on Copula-HDP-HMM shows minimal I/O overhead over baselines that suffer from a suboptimal grouping of the underlying count vectors that affects the predictive performance, and consequently the hitrates and the I/O efficiency. For instance, for MS1, HULK loads only about a fourth of the data compared to its baseline.
– We have evaluated HULK on a diverse set of real world benchmark traces where it showed improved hitrates with reduced I/O overheads offering a practical strategy for Bulk Cache Preloading.

7 Summary and conclusions

We propose Copula-HDP-HMM, a Bayesian non-parametric model for temporal multivariate data with arbitrary marginals, addressing a fundamental machine learning problem, applicable beyond caching. Inference is hard for non-continuous marginals. We propose efficient inference based on extended rank likelihood in a DP-mixture setting. Finally, we propose HULK, a novel technique based on Copula-HDP-HMM, for I/O

efficient bulk cache preloading, that shows significant hitrate improvements and an order of magnitude I/O efficiency improvement. We also hope our work paves the way for further research on new machine learning algorithms for new forms of data from emerging fields like bulk cache preloading.

References

[1] B. S. Gill and D. S. Modha, "Sarc: Seq prefetching in adaptive replacement cache," in *UATC*, 2005.

[2] M. M. Gokul Soundararajan and C. Amza, "Context-aware prefetching at the storage server," in *USENIX ATC*. USENIX, 2008.

[3] C. B. Lavanya Tekumalla, "Mining block i/o traces for cache preloading with sparse temporal non-parametric mixture of multivariate poisson," in *SDM*, 2015.

[4] K. Kawamura, "The structure of multivariate poisson distribution," *Kodai Mathematical Journal*, 1979.

[5] P. Tsiamyrtzis and D. Karlis, "Strategies for efficient computation of multivariate poisson probabilities," *Communications in Statistics (Simulation and Computation)*, vol. Vol. 33, No. 2, pp. 271292, 2004.

[6] R. V. Ray. M., "Gaussian process conditional copulas with applications to financial time series," in *NIPS*, 2013.

[7] A. S. Benjamin Letham, Wei Sun, "Latent variable copula inference for bundle pricing from retail transaction data," in *ICML*, 2014.

[8] R. V. Ray. M., "Copula mixture model for dependency-seeking clustering," in *ICML*, 2012.

[9] D. Karlis and A. Nikoloulopoulos, "Modelling multivariate count data using copulas," *Communications in Statistics - Simulation and Computation*, 2009.

[10] M. S. Smith and M. A. Khaled, "Estimation of copula models with discrete margins via bayesian augmentation," in *Journal of Amarican Stat Association*, 2012.

[11] P. Hoff, "Extending the rank likelihood for semiparametric copula estimation," in *Annals of Applied Statistics*, 2007.

[12] A. Sklar, "Fonctions de repartition 'a n dimensions et leurs marges," *Publications delInstitut Statistique de lUniversite de Paris*, vol. Vol 8, 229-231 (1959), 2004.

[13] H. Joe, "Dependence modeling with copulas."

[14] A. M. Schmidt and M. A. Rodriguez, "Modelling multivariate counts varying continuously in space," in *Bayesian Statistics Vol 9*, 2010.

[15] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of American Statistical Association*, 2004.

[16] M. I. J. Emily B. Fox, Erik B. Sudderth and A. S. Willsky, "A sticky hdp-hmm with application to speaker diarization," *The Annals of Applied Statistics*, 2011.

[17] E. Fox, E. Sudderth, M. Jordan, and A. Willsky, "A Sticky HDP-HMM with Application to Speaker Diarization," *Annals of Applied Statistics*, 2011.

[18] R. M. Neal, "Markov chain sampling methods for dirichlet process mixture models," in *Journal of Computational and Graphical Stats*. USENIX, 2000.

[19] C. E. Rasmussen, "The infinite gaussian mixture model," in *NIPS*, 1999.

[20] Y. Zhang, G. Soundararajan, M. W. Storer, L. N. Bairavasundaram, S. Subbiah, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Warming up storage-level caches with bonfire," in *USENIX FAST*, 2013.

[21] C. Z. S. S. M. Li, Z. and Y. Zhou, "Mining block correlations in storage systems," in *FAST*, 2004.

[22] Y. Shahar, "Shahar. a framework for knowledge-based temporal abstraction." *Artificial Intelligence*, 1997.

[23] I. Batal, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht, "Mining recent temporal patterns for event detection in multivariate time series data," *KDD*, 2012.

[24] S. K. S. Q. L. D. D. N. H. Y. K. T. Tu Bao Ho, Trong Dung Nguyen, "Mining hepatitis data with temporal abstraction." *KDD*, 2003.

[25] A. R. Dushyanth Narayanan, A Donnelly, "Write off-loading: Practical power management for enterprise storage," *USENIX, FAST*, 2008.

Appendix

We briefly describe our dynamic programming based prediction algorithm (alg 3). (More details in suppl material).

Algorithm 3: **Prediction Algorithm:** The input to the prediction algorithm comprises of a sequence of state IDs for the live trace $\bar{\mathcal{Z}}_L = (\bar{Z}_1, \ldots, \bar{Z}_{T_{live}})$ a sequence of state IDs for the repository trace $\mathcal{Z}_R = (Z_1, \ldots, Z_{T_{repo}})$. The algorithm finds best locally aligned subsequence of live trace $\mathcal{Z}_L$, with the sequence repo sequence in its entirity $\mathcal{Z}_R$. It returns the best preload candidates as a set $P \subset [T_{live}]$ of slice indices, containing a window of W slices in the repository around the alignment point. Positional weights are assigned with the weighing function (suppl material) since alignment at end of trace is more significant than beginning. Note, S is a cluster similarity matrix derived from the distance between cluster means. (Note: We use $T_{live} = 50$ and W=5).

Input : $\bar{\mathcal{Z}}_L = \{\bar{Z}_1, \ldots, \bar{Z}_{T_{live}}\}, \mathcal{Z}_R = \{Z_1, \ldots, Z_{T_{repo}}\}$, $S \in \mathcal{R}^{K \times K}$

Output : set $P \subset [T_{live}]$

$M[s,t] = 0, \forall s \in \{1, \ldots, T_{repo}\}, t \in \{1, \ldots, T_{live}\}$

for $s \in \{1, \ldots, T_{repo}\}$ **do**

  for $t \in \{1, \ldots, T_{live}\}$ **do**

$$obj[1] = (M[s-1, t-1] + S(Z_{s-1}, \bar{Z}_{t-1}) * wfunc(t, N_R)) \quad \text{Match}$$

$$obj[2] = M[s, t-1] - \delta \quad \text{Insert}$$

$$obj[3] = M[s-1, t] - \delta \quad \text{Delete}$$

$$obj[4] = 0 \quad \text{Restart}$$

$M[s,t] = Max(obj)$

Switch($argMax(obj)$)

Case Match: $Prev(s,t) = (s-1, t-1)$

Case Insert: $Prev(s,t) = (s, t-1)$

Case Delete: $Prev(s,t) = (s-1, t)$

Case Restart: $Prev(s,t) = (s,t)$

$\hat{t} = T_{live}$; $\hat{s} = ArgMax(M[:, T_{live}])$

return : $P = \{s+1, \ldots, s+1+W\}$