Host Load Prediction with a Bayesian Model

based on the paper ”Host Load Prediction in a Google Compute Cloud with a Bayesian Model” by Sheng Di, Derrick Kondo, Walfredo Cirne INRIA, France, Google Inc., USA sheng.di,derrick.kondo@inria.fr, walfredo@google.com

David Herzog-B., Johannes Priewasser

May 23, 2013

Why do we need host load prediction in the cloud?

- essential for achieving SLA's
- challenging because it fluctuates drastically at small timescales
- prediction of host load enables
  - proactive job scheduling
  - host load balancing decisions
  - improve resource utilization
  - lower data center costs
  - improve job performance

Why do not use existing methods for the cloud?

- more challenging than traditional Grids and HPC systems
- differences in the workloads run on top of such platforms
- tasks tend to be shorter and more interactive
- Google data center compared to AuverGrid cluster has $[\frac{1}{20}, \frac{1}{2}]$ task length
- leads to much finer resource allocation on Googles data centers

Prior work

- prior prediction work in Grid Computing or HPC systems has focused on
  - using moving averages,
  - auto-regression and
  - noise filters
- applied to bursty Cloud workloads, they have limited accuracy
- does not attempt to predict long-term future load over consecutive time intervals
- filtering the noise of host load in Clouds may remove important and real fluctuations

The expectations of using host load prediction with bayesian model

- prediction method based on Bayes model to predict
  - mean load over a long-term time interval (up to 16 hours)
  - mean load in consecutive future time intervals (pattern)
- predictive features that capture
  - expectation,
  - predictability,
  - trends and
  - patterns of host load

The expectations of using host load prediction with bayesian model

- predict host load over a longterm period up to 16 hours
- two critical metrics:
  - CPU
  - memory
- use a Bayesian model for prediction
- effectively retains the important information about load fluctuation and noise

The results of using host load prediction with bayesian model

- evaluated using detailed one-month trace of a Google data center
- method achieves high accuracy with a mean squared error of
  - 0.0014 for a single interval
  - $\leq 10^{-5}$ for a pattern
- improves load prediction accuracy by 5.6-50% compared to other methods

Google load measurements, the databasis...

- google data center, production system of 12,000 machines
- traced over 670,000 jobs and over 40 million task events over one month
- users submit jobs to a batch scheduler
- each job consists of a set of tasks and a set of resource constraints
- batch scheduler allocates those tasks to hosts
- load on the hosts is a function of the incoming workload at the batch scheduler, and its scheduling strategy

Google load measurements

- host load at a given time point is the total load of all running tasks on that particular machine
- calculate the relative load values by dividing the absolute load values by the corresponding capacities
- load values range between 0 and 1 for each resource
- discretize all the load values by recomputing each hosts relative load over consecutive fixed-length periods, each having length on the order of a few minutes
- discretized load trace is the basis of the paper

Google load measurements

- compute the relative host load at each period for AuverGrid's trace, using the same method applied to the Google trace
- higher noise compared with Auver-Grid
- minimum/mean/maximum noise of CPU load over all hosts computed using a mean filter are for
  - AuverGrid: 0.00008, 0.0011, 0.0026
  - Google: 0.00024, 0.028, 0.081

Google load measurements

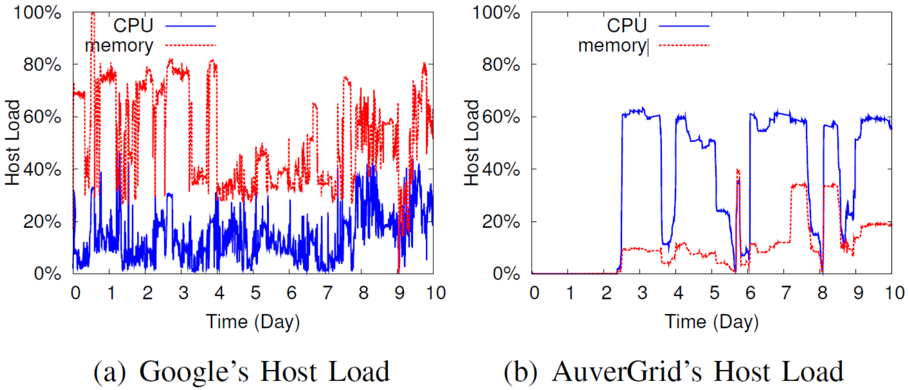
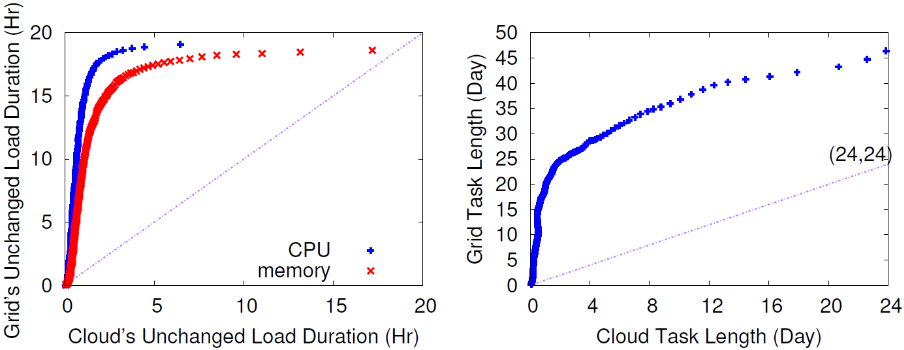


Figure : Load Comparison between Google & AuverGrid

Google load measurements

- compare the distributions of host load using a quantile-quantile plot
- equally split the range of load values into five sub-ranges
- load duration is defined to be the length of time where load on a host constantly stays within a single sub-range
- compare the distribution of these load durations
- figure shows the points at which the load durations have the same probability

Google load measurements



(a) QQPlot of Unchanged Duration (b) QQplot of Task Length

Figure : Quantile-Quantile Plot of Continuous Duration and Task Length

Prediction formulation

- predict the mean load over a single interval at a current time point t_0
- predict the mean load over consecutive time intervals
- introduce **exponentially segmented pattern (ESP)**, to characterize the host load fluctuation over a time period
- prediction interval split into a set of consecutive segments, whose lengths increase exponentially
- predict the mean load over each time segment

Prediction formulation

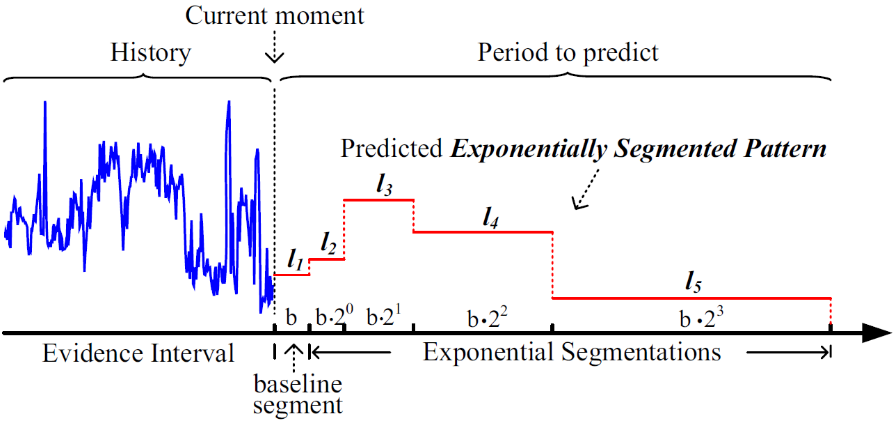


Figure : Illustration of Exponentially Segmented Pattern

Prediction formulation

- total prediction interval length s
- first segment (s_1) is called baseline segment with length b, starts from the current time point t_0 and ends at $t_0 + b$
- length of each following segment (s_i) is $b * 2^{i-2} | i = 2, 3, 4, ...$
- if b is set to 1 hour, the entire prediction interval length s is 16 (=1+1+2+4+8) hours
- predict mean host load for each segment
- mean values are denoted by $l_i | i = 1, 2, 3, ...$
- prediction granularity is finer in the short-term than in the long-term

Prediction formulation

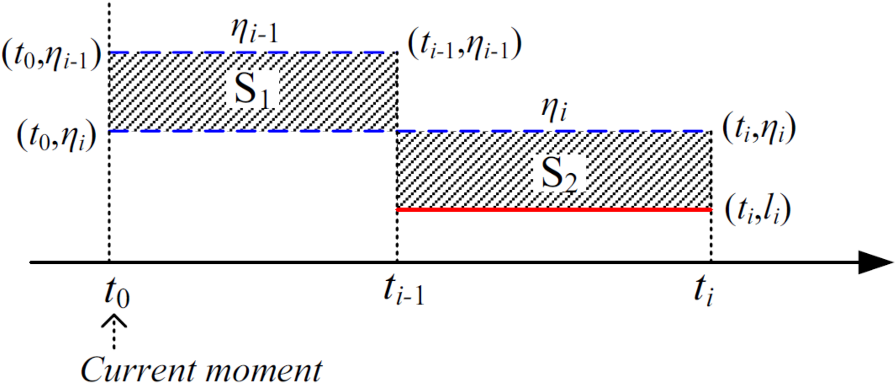


Figure : Induction of Segmented Host Load

Prediction formulation

- predict the vector of load values ($l = (l_1, l_2, ..., l_n)^T$) where each value represents the mean load value over a particular segment
- a predictor often uses recent load samples
- interval that encloses the recent samples is called evidence interval or **evidence window**
- prediction of each segmented mean load is the key step

Prediction formulation

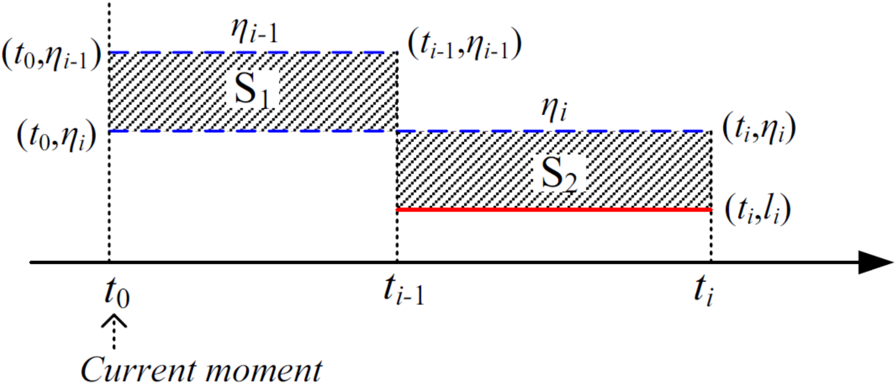


Figure : Induction of Segmented Host Load

Prediction formulation

- host load has high correlation between adjacent short-term intervals but not for the non-adjacent ones
- it is straight-forward to predict the load in the successive intervals based on the evidence window
- convert the segment representation into another one, in which each interval to be predicted is adjacent to the evidence window
- need to predict a set of mean host loads for different lengths of future intervals, each starting from the current time t_0
- mean load levels of the prediction intervals as $\eta_1, \eta_2, \ldots, \eta_n | \eta_{i+1} = 2\eta_i$
- target is to predict such a vector, $\eta = (\eta_1, \eta_2, \ldots, \eta_n)^T$, rather than the vector l

Prediction formulation

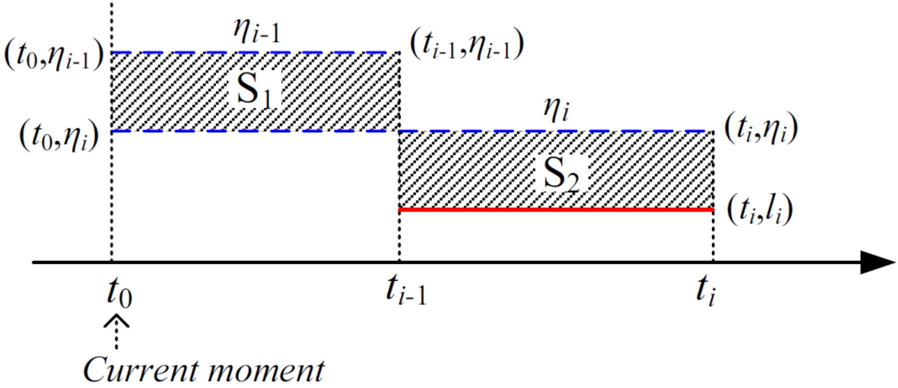


Figure : Induction of Segmented Host Load

Prediction formulation

- vector l can be converted from η through the following induction
- current moment is t_0
- already predicted two mean load values (η_{i-1} and η_i) over two intervals, $[t_0, t_{i-1}]$ and $[t_0, t_i]$
- making the areas of the two shaded squares (S_1 and S_2) equal to each other
- derive the mean load value in $[t_{i-1}, t_i]$
- l_i is the predicted mean load in the new segment $[t_{i-1}, t_i]$, corresponding to the red solid-line segment

$$l_i = \eta_i + \frac{t_{i-1} - t_0}{t_i - t_{i-1}}(\eta_i - \eta_{i-1}) \tag{1}$$

Prediction formulation

- Taking into account $t_i = 2t_{i-1}$ and $t_0 = 0$ the formula simplifies to

$$l_i = 2\eta_i - \eta_{i-1} \tag{2}$$

- simplifies and generalizes predictor implementation as any predictor that can predict load over a single load interval can be converted to predict a load pattern
- gives the option of predicting different load intervals starting at the current time point, or consecutive load intervals, without any additional overheads

Prediction formulation

- 2 key steps
  - mean load prediction (lines 1-5)
  - segment transformation (line 6)
- each prediction interval always starts from the current moment, unlike the segments defined in the first representation *I*

Prediction formulation

Algorithm 1 PATTERN PREDICTION ALGORITHM

Input: baseline interval (b); length of prediction interval ($s = b \cdot 2^{n-1}$, where n is the number of segments to be split in the pattern prediction);
Output: mean load vector l of Exponentially Segmented Pattern (ESP)

1: **for** ($i = 0 \to n-1$) **do**
2: $z_i = b \cdot 2^i$;
3: $\varpi_i = \frac{z_i}{2}$; /*ϖ_i is the length of the evidence window*/
4: Predict the mean load η_i, whose prediction length is equal to z_i, based on a predictor - PREDICTOR(ϖ_i,z_i);
5: **end for**
6: Segment transformation based on Equation (2): $\boldsymbol{\eta} \to \boldsymbol{l}$;

Figure : pseudo-code of Cloud load pattern prediction method

Prediction formulation

- shortterm prediction error always lags behind long-term error
- short-term prediction error almost always lags behind the long-length prediction error
- lag time is close to the interval length

Prediction formulation

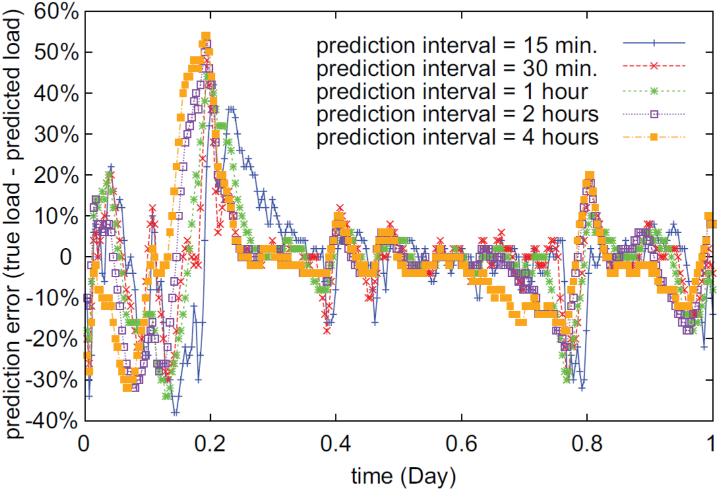


Figure : Prediction Errors in Different Prediction Lengths

Bayes model

- generate
  - the posterior probability from the prior probability distribution
  - the run-time evidence of the recent load fluctuations
- according to a Bayes Classifier
- bayes classifier is a classic supervised learning classifier used in data mining

Bayes model

Bayesian classification consists of five main steps:

1. - determine the set of target states ($W = (\omega_1, \omega_2, ..., \omega_m)^T$, where m is the number of states)
   - evidence vector with h mutually-independent features ($\chi = (x_1, x_2, ..., x_h)^T$)
2. compute the prior probability distribution for the target states, $P(\omega_i)$, based on the samples
3. compute the joint probability distribution $p(\chi|\omega_i)$ for each state ω_i
4. compute the posterior probability based on some evidence

$$P(\omega_i|x_j) = \frac{p(x_j|\omega_i)P(\omega_i)}{\sum_{k=1}^{m} p(x_j|\omega_k)P(\omega_k)} \tag{3}$$

5. make the decision based on a risk function $\lambda(\hat{\omega}_i, \dot{\omega}_i)$, where $\hat{\omega}_i$ and $\dot{\omega}_i$ indicate the true value and predicted value of the state

Bayes model

- Based on different risk functions, there are two main ways for making decisions
  - Naive Bayes Classifier (N-BC)
  - Minimized MSE (MMSE) based Bayes Classifier (MMSE-BC)
- Their corresponding risk functions are:

$$\lambda(\dot{\omega}_i, \hat{\omega}_i) = \begin{cases} 0 & |\dot{\omega}_i - \hat{\omega}_i| < \delta \\ 1 & |\dot{\omega}_i - \hat{\omega}_i| \geq \delta \end{cases} \tag{4}$$

$$\lambda(\dot{\omega}_i, \hat{\omega}_i) = (\dot{\omega}_i - \hat{\omega}_i)^2 \tag{5}$$

Bayes model

- predicted value of the state ($\hat{\omega}_i$) is determined by:
  minimal error:
  $$\hat{\omega}_i = \arg\max p(\omega_i | x_j) \tag{6}$$
  MSE:
  $$\hat{\omega}_i = \sum_{i=1}^{m} \omega_i p(\omega_i | x_j) \tag{7}$$
- the target state vector and the evidence feature vector are the most critical for accurate prediction
- split the range of host load values into small intervals, and each interval corresponds to a usage level
- number of intervals in the load range [0,1] is denoted by r
- 50 load states in total, $[0, 0.02), [0.02, 0.04), \ldots, [0.98, 1]$

Bayes model

- the length of the evidence window is set equal to half of the prediction interval length
- maximizes accuracy, based on experimental results
- evidence window will also be split into a set of equally-size segments
- if prediction interval length is 8 hours, the evidence window length will be set to the recent past 4 hours
- 48 ($= \frac{4*60}{5}$) successive load values (sample interval 5 minutes) in this period will serve as the fundamental evidence
- prediction interval length is 4 hours, evidence window length is 2 hours and there are 24 load values in the evidence window

Bayes model

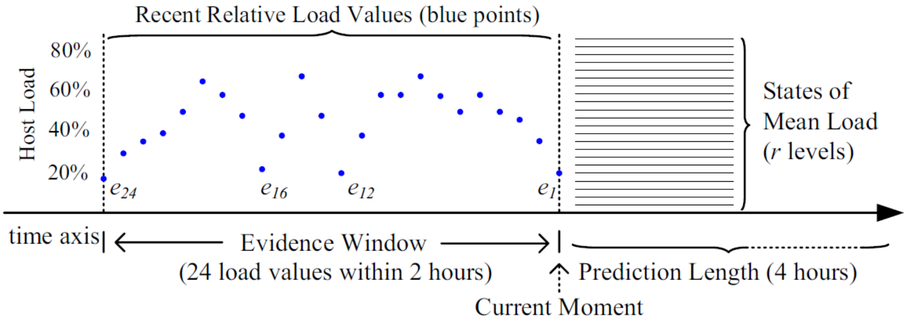


Figure : Illustration of Evidence Window and Target Load States

Features of Load Fluctuation

- 9 candidate features to be used as the evidence
- load vector in the evidence window $e = (e_1, e_2, ..., e_d)^T$ where d is the number of the samples in the evidence window, also known as window size
- elements in the vector are organized from the most recent one to the oldest one

mean load ($F_{ml}(e)$)

- mean load is the mean value of the load vector e

$$F_{ml}(e) = \frac{1}{d}\sum_{i=1}^{d} e_i \tag{8}$$

- the value range of the mean load is [0,1] in principle
- split this range into r even fractions each corresponding to a load level
- this features value must be one of the 50 levels

weighted mean load ($F_{wml}(e)$)

- weighted mean load refers to the linear weighted mean value of the load vector e

$$F_{wml}(e) = \frac{\sum_{i=1}^{d}(d-i+1)e_i}{\sum_{i=1}^{d} i} = \frac{2}{d(d+1)} \sum_{i=1}^{d}(d-i+1)e_i \qquad (9)$$

- the weighted mean load weights the recent load values more heavily than older ones
- this feature is also within [0,1], which will also be split into 50 levels to choose

fairness index ($F_{fi}(e)$)

- The fairness index is used to characterize the degree of the load fluctuation in the evidence window

$$F_{fi}(e) = \frac{(\sum_{i=1}^{d} e_i)^2}{d \sum_{i=1}^{d} e_i^2} \tag{10}$$

- Its value is normalized in [0,1]
- a higher value indicates more stable load fluctuation
- Its value is equal to 1 if and only if all the load values are equal to each other
- Since the target state in our model is the mean load value of the future prediction interval, the mean load feature seems more important than the fairness index
- when the load in the prediction interval changes with the similar fluctuation rule to the statistics, fairness index could effectively improve the prediction effect

noise-decreased fairness index ($F_{ndfi}(e)$)

- also computed using fairness index formula
- if there exist one or two load values (load outliers) that may significantly degrade the whole fairness index, they would not be counted in
- such load outliers are likely supposed to be considered noise or irregular jitters

type state ($F_{ts}(e)$)

- The type state feature is used to characterize the load range in the evidence window and the degree of jitter
- defined as a two-tuple, α, β, where α and β refer to the number of types involved and the number of state changes
- if the window vector is (0.023,0.042,0.032,0.045,0.056,0.036)
- then there are only two types (or levels) involved, [0.02,0.04) and [0.04,0.06)
- there are four state changes: $0.023 \rightarrow 0.042, 0.042 \rightarrow 0.032, 0.032 \rightarrow 0.045, 0.056 \rightarrow 0.036$
- 0.056 is not a state change since its preceding state is at the same level

first-last load ($F_{fll}(e)$)

- The first-last load feature is used to roughly characterize the changing trend of the host load in the recent past
- defined as a two-tuple, τ, υ indicating the first load value and the last one recorded in the evidence window
- rough feature which needs to be combined with other features in practice

N-segment pattern ($F_{N-sp}(e)$)

- characterize the segment patterns based on the evidence window
- evidence window is evenly split into several segments, each of which is reflected by the mean load value
- if the window length is 4 hours (i.e., the window size is 48), then the 4-segment pattern is a fourtuple, whose elements are the means of the following load values respectively, $[e_1, e_{12}], [e_{13}, e_{24}], [e_{25}, e_{36}], [e_{37}, e_{48}]$
- N is set to 2, 3, and 4 respectively, so there are actually three features w.r.t the N-segment pattern

feature correlation

- some of the features are mutually correlated, which violates the assumption of feature independence in Bayes theorem
- features used in Formula (3) should be mutually independent
- the fairness index feature and the noise-decreased fairness index feature could be closely correlated
- list of linear correlation coefficients and Spearmans rank correlation coefficients

feature correlation

	F_{ml}	F_{wml}	F_{fi}	F_{ndfi}	F_{ts}	F_{fll}	$F_{N\text{-}sp}$
F_{ml}	N	N	Y	Y	Y	Y	N
F_{wml}	N	N	Y	Y	Y	Y	N
F_{fi}	Y	Y	N	N	Y	Y	Y
F_{ndfi}	Y	Y	N	N	Y	Y	Y
F_{ts}	Y	Y	Y	Y	N	Y	Y
F_{fll}	Y	Y	Y	Y	Y	N	Y
$F_{N\text{-}sp}$	N	N	Y	Y	Y	Y	N

Figure : compatibility of the features

feature correlation

- Since there are 9 features ($F_{ml}, F_{wml}, F_{fi}, F_{ndfi}, F_{ts}, F_{fll}, F_{2-sp}, F_{3-sp}, F_{4-sp}$) in total, the number of their combinations is at most 2^9
- classified into 4 groups, $\{F_{ml}, F_{wml}, F_{2-sp}, F_{3-sp}, F_{4-sp}\}, \{F_{ndfi}, F_{fi}\}, \{F_{ts}\}, \{F_{fll}\}$
- elements in the same group cannot be used meanwhile in one combination
- numbers of compatible combinations (NCC) are 6, 3, 2, 2
- NCC(9 features) = NCC(Group 1) * NCC(Group 2) * NCC(Group 3) * NCC(Group 4) = 6 * 3 * 2 * 2 = 72
- excluding the case where no feature is selected
- 71 viable combinations of the features

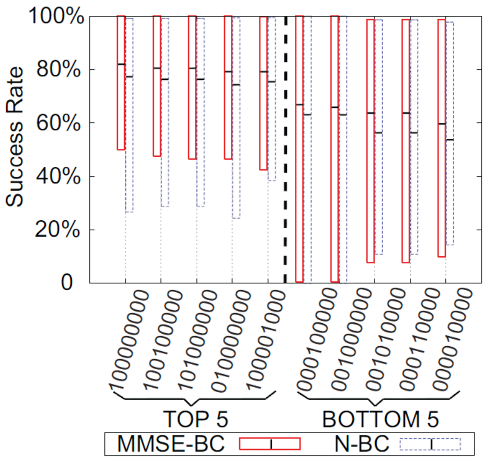
Experimental Results

- split one-month trace data into two durations:
  - training period (beginning - 25th day)
  - test period (26th day - end)
- training period is used to fit the models, e.g. for computing the prior probability $P(\omega_i)$ and the conditional probability $p(x_j|\omega_k)$

Experimental Results

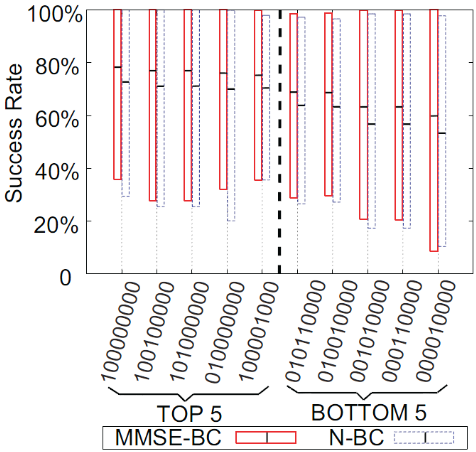
- best feature combination is always 100000000 F_{ml}, worst one is always 000010000 F_{ts} regardless of the prediction interval length
- prediction of MMSE-BC is always more accurate than that of N-BC
- MMSE-BC adopts the mathematically expected value of the predicted load, which has the highest probability of being located in the real load level
- N-BC selects the level with the highest posterior probability as the prediction result

Experimental Results



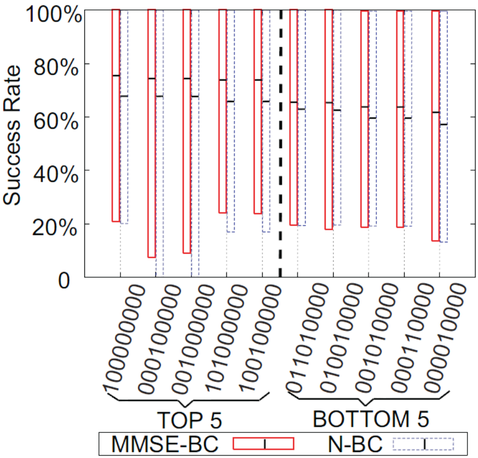
(a) Success Rate (s=3.2h)

Experimental Results



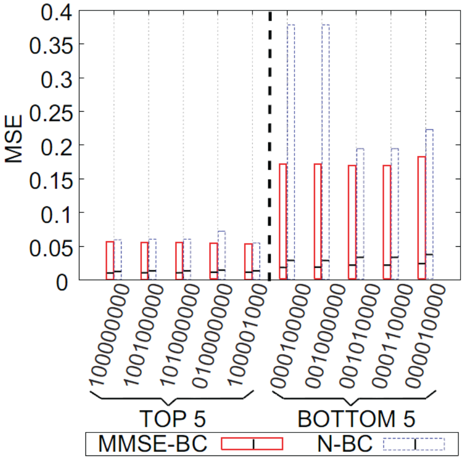
(c) Success Rate (s=6.4h)

Experimental Results



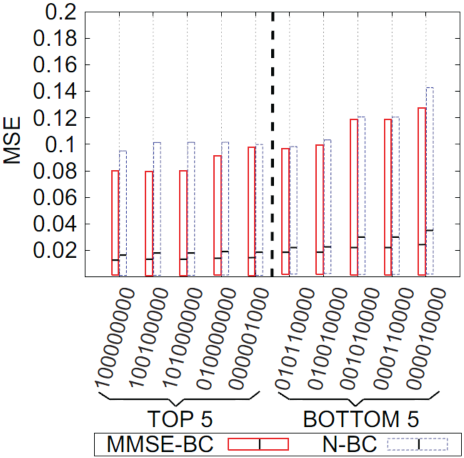
(e) Success Rate (s=12.8h)

Experimental Results



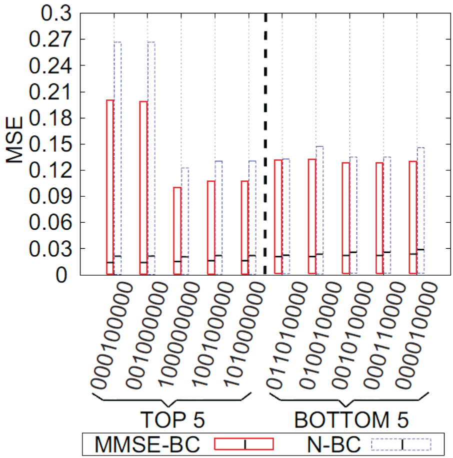
(b) MSE (s=3.2h)

Experimental Results



(d) MSE (s=6.4h)

Experimental Results



(f) MSE (s=12.8h)

end

Thank you for your attention

Referenced paper:

Host Load Prediction in a Google Compute Cloud with a Bayesian Model
SC12, November 10-16, 2012, Salt Lake City, Utah, USA
978-1-4673-0806-9/12/$31.00 ©2012 IEEE
by Sheng Di, Derrick Kondo, Walfredo Cirne INRIA, France, Google Inc., USA
sheng.di,derrick.kondo@inria.fr, walfredo@google.com