IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 November 2023, accepted 23 November 2023, date of publication 8 December 2023,
date of current version 20 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3341046

RESEARCH ARTICLE

Dependency Prediction of Long-Time Resource Uses in HPC Environment

NAVIN MANI UPADHYAY[1], RAVI SHANKAR SINGH[1], (Senior Member, IEEE), AND SHRI PRAKASH DWIVEDI[2]

[1]Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, Varanasi, Uttar Pradesh 221007, India
[2]Department of Information Technology, G. B. Pant University of Agriculture and Technology, Pantnagar 263145, India

Corresponding author: Navin Mani Upadhyay (navinmupadhyay.rs.cse17@iitbhu.ac.in)

ABSTRACT High-Performance computing provides a new infrastructure for scientific calculation and its simulation. However, unbalanced load distribution among the processors causes a decreased performance in such computations, and creates a massive requirement of computing resource allocation, that requires an increased simulation. Therefore long-range resource utilization prediction becomes essential to achieve optimal performance in an HPC environment. This paper introduces a novel ensemble technique, which includes two algorithms, the Feature-based capability prediction algorithm(FBCA), and the Accuracy and Relative Runtime Error Prediction Algorithm (ARRE). A three-level architectural framework (the simulation environment, resource prediction, and resource queue) has also been proposed and tested on Phold and SoS. The proposed framework can deal with the requirements of computing and simulations. The FBCA algorithm reduces the redundancy between available features, and the ARRE algorithm ensures our ensemble technique's effectiveness. We have compared the performance of the proposed schemes with other existing methods such as the Regressive Approach, Linear Regression and Random Forest, and found that our proposed algorithm achieves better accuracy from 8% to 18%.

INDEX TERMS Multi-core processors, resource prediction, social opinion system, high-performance computing, parallel and discrete simulation environment.

I. INTRODUCTION

In today's computational era, the demand for Scientific Computing is increasing day-by-day because scientific computations affect almost every area of our life, such as science and engineering, society and economy, and many other untouched areas. Therefore the requirements of accuracy and new calculation methodology become important. Such systems have many interacting components, so they are categorized as complex systems. The study of complex systems and their simulation has become a powerful task for the new generation of researchers, and the results are obtained in the form of different supercomputers. One of them is Param Shivay Supercomputer,[1] designed by CDAC Pune, India. This supercomputer is established at Indian Institute of Technology (Banaras Hindu University) IIT(BHU), under the National Supercomputing Mission (NSM) shown in Table-1.

TABLE 1. Assumptions for parallel computing architectures.

System Specifications	Available Resources
Theoretical Peak-Floating-Point performance Total (Rpeak)	837 TFLOPS
Sustained Performance (Rmax)	425 TFLOPS(CPU only nodes) + 100 TFLOPS (GPU Nodes)
Base specifications (Compute nodes)	2 x Intel Xeon Skylake, 6148, 20 Cores, 2.4 GHz, processors/node, 192 GB memory, 480 GB SSD
Master/Service/Login Nodes	10 Nos.
CPU only compute Nodes (Memory)	192 Nodes (192 GB each)
High Memory Nodes	20 Nos. (7680 GB)
GPU compute Nodes	11 (22 Nvidia V100PCIe)
Interconnect	Primary: 100 Gbps, Mellanox Infiniband.

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Alawneh.

[1]https://nsmindia.in/node/155

In the above table-1, it has shown that the Param Shivay supercomputer at IIT(BHU) Varanasi, boasts impressive system specifications that underscore its computational prowess. The theoretical peak-floating-point performance, a staggering 837 TFLOPS, showcases its capacity for complex calculations. In terms of sustained performance, the supercomputer achieves a remarkable 425 TFLOPS, combining the power of CPU-only nodes (425 TFLOPS) and GPU nodes (100 TFLOPS). The base specifications of the compute nodes reveal a robust architecture. Each node features 2 x Intel Xeon Skylake processors with 20 cores each, clocking at 2.4 GHz. With 192 GB of memory and a 480 GB SSD, these nodes form the backbone of Param Shivay's computational muscle. The primary, service, and login nodes, numbering 10, facilitate efficient management and user interactions. Diving into the specialized nodes, the CPU-only compute nodes, totaling 192, each equipped with 192 GB of memory, contribute to the system's parallel processing capabilities. Additionally, 20 high-memory nodes, boasting a substantial 7680 GB of memory, cater to memory-intensive tasks. The inclusion of 11 GPU compute nodes, housing 22 Nvidia V100PCIe GPUs collectively, underscores the supercomputer's versatility in handling parallel tasks and accelerating specific workloads. The interconnect infrastructure plays a crucial role, with a primary 100 Gbps Mellanox Infiniband ensuring seamless communication between nodes.

The computation power of supercomputers brings a different perspective to the research and overcomes many traditional computing limitations [2], [6], [8]. Examples of such limitations are non-linear relationship modeling, solving random problems, solving long-iterative social evolution problems and algorithms, etc. Nowadays, the scale of such simulation applications with these traditional limitations is also becoming complex, so a parallel discrete event simulation has come into the picture, which provides high-performance solutions for such problems. The simulation applications vary in their simulation environment and scenarios. A high-performance computing environment provides a platform to support and control the scale of such simulations. For such simulations, the execution environment for all scientific tasks is usually well-defined before execution because of its dependency on specific libraries, applications, job schedulers, and the currently running operating systems. Thus those who use such services are bound to use limited implementation of specific applications. In this context, the operator teams will only be able to manage or control the extensions of the running time environment. They will also be able to control the cooperation of system resources, generally available at the processor level. Therefore access to the HPC resources will become more restricted, and all the jobs will have to wait for their execution in a priority queue. Conversely, this may happen because most of the physical machines are under-utilized.

These limitations overcome a high-demanding High Performance Computing Clusters (HPCC) model only when the HPCC tool applies the cloud concept to the existing resources. The result will be the Infrastructure as a service (IaaS) for its delivery model [1], [3], [4], [15], [16]. The cloud computing approach promises to increase flexibility and efficiency in energy consumption and cost reduction. Such providers with a large infrastructure and huge amounts of resources can only gain a good economic scale. The public IaaS resources may be better in utilization than the private ones. Therefore, this study focuses on large-scale High-performance computing simulation based on parallel discrete event simulation on Param Shivay Supercomputer.

A. MOTIVATION

A lot of different methods and models have been introduced by several researchers. These works are based on the tasks' decomposition and historical tracing. The decomposition of task (DT) model will first decompose the assigned tasks into small individual tasks and then predict the required resources. For prediction, it measures the requirement of each task. The historical trace (HT) model predicts the required resources based on the nature of jobs and their execution patterns. However, some researchers have claimed that one cannot accurately predict the resources needed for running applications in an HPC environment [2], [3], [5], [40], [42]. These claims motivate us to work on dependency prediction of long-term resource utilization.

B. CONTRIBUTION

To solve the above discussed problems, we have proposed a novel computing approach based on the ensemble technique to predict the long-term required resources in the HPC environment. The following contribution has been made by us in this paper:

- A novel Ensemble Algorithm has been proposed with multiple objectives.
- A new algorithm has been proposed to predict a valid sequence of execution of jobs.
- A resource prediction framework has been proposed. This framework describes the simulation models which computes the required resources and predicts the job queue of execution.
- A feature-based capability prediction algorithm (FBCA) with its accuracy and relative error prediction algorithm to improve its performance is also proposed.
- The proposed work has been validated based on several workflow scheduling algorithms.
- A sequential and parallel execution of (Uniform and random based data selection) SVM, KNN, DT, Naive Bayes and Gradient Boosting algorithms has demonstrated based on different number of threads.
- To show the effectiveness of our proposed work, a statistical method ''Fired-man Test'' has been demonstrated.

C. PAPER ORGANIZATION

The rest of the paper is organized as follows. Section II describes a review of related works and algorithms.

Section III presents a basic understanding of the requirements, Notation used, assumptions and basic definitions. Section IV describes problem formulation. The Section V presents a detailed discussion of the models used in the proposed algorithms with its framework. In Section VI a standard overview of work-flow scheduling and resource prediction algorithms has been introduced in terms of the experimental setup and discussion. Section VII discusses the proposed algorithm and its relative performance prediction in terms of results and discussion followed by statistical analysis. In Section VII we have analyzed and validated the proposed algorithms on different scientific workflow scheduling algorithms. Section IX presents the conclusion and future work.

II. RELATED WORK

Based on the research gaps, we found that many researchers have already done some work. Most of the work refers to the domain of Multi-core, GPUs and CPUs to use instances and other services. Some have worked on an architecture-based system that reflects grids and clusters [6], [8], [17], [18]. The maintenance cost of grid and clusters are very high, and computing resources are also not easily available. Once the computing resources are available, the user can require an enormous cost with large equipment and space to install them. So, in this situation, computing the long-time running resources will give a significant change in execution time for any scientific computing problem.

To provide a better execution time, researchers have proposed work [1], [7] in the domain of data parallelization, load balancing, cloud-based solutions, etc. This paper is focused on three aspects, i.e., mining on CPU Datasets, performance prediction, and workload characterization. A detailed review of the work related to the above is presented as follows.

A. JOB'S RUNNING TIME PREDICTION

In Scheduling algorithms, the accuracy of job scheduling prediction is an important and influential area of research. Several researchers have proposed strategies to predict a HPC Job's execution time. A fitting model for user's estimation has been proposed by Xiaomeng et al. [4]. He and his team have prioritized users' habits for their estimation. The drawback of their proposed algorithm is that their experiments' prediction accuracy is not ideal and uniform.

A new loss function has been proposed by Hamidreza et al. [8]. They investigate and claimed that their proposed method is easy and effective for backfill scheduling experiments. The demerits of their proposed model are based on scalability because they have used Last Level Cache (LLC), Dynamic Random Access Memory (DRAM), PCI-E links only, which is not friendly for other NUMA architectures benchmarks. In LLC case the demerit is when we scale up the number of cores in a processor, the shared last-level cache can become a bottleneck. For example: Suppose A quad-core processor has a shared LLC of 8 MB. Each core has equal access to this cache. However, as one user move to a 16-core processor while keeping the same 8 MB LLC, each core now has to share the cache with more peers, potentially leading to increased contention and reduced performance per core. Similarly, The demerit in scalability for DRAM is often seen in the increasing latency and limited bandwidth as the number of processors or cores grows. For example: Consider that we have a server with 32 GB of DDR4 RAM, and it provides a certain level of bandwidth. If we double the number of processors without increasing the memory capacity or bandwidth, each processor now has to contend for access to the same memory resources, leading to potential slowdowns. The scalability issues with PCI-E links arise when multiple devices, such as GPUs or storage controllers, share the same PCI-E bus. This can lead to contention for bandwidth. For example: Consider a system with a single GPU connected via PCI-E, providing a certain bandwidth. If we add another GPU or high-bandwidth device without expanding the PCI-E infrastructure, both devices now have to share the available bandwidth, potentially reducing the performance benefits of the additional hardware.

One more method to predict the maximum likelihood estimation has been proposed by Richard et al. [9]. They used a liner regression model to compare their result with other's existing methods. They found a good improvement of $\approx 22\%$.

Wang et al. [21] predicted the job's runtime based on the K-NN algorithm. The demerit of their proposed method is when the data distribution is scattered they won't have any ideal prediction in this case. One more interesting approach has been proposed by Mejri et al. [56]. They have studied skiers from a certain time to predict the result using an ensemble technique. Phansalkar et al. [10] have used their result based on three different algorithms i.e. decision tree, random forest and KNN. They found that the decision tree produces the best set of prediction results. In recent years, it has been found that ensembling multiple learning algorithms is often better than using only a single learning algorithm.

HPC applications come up with many logical processes, such processes always require simulation entities. So, to simulate HPC applications in a virtual environment Shaw et al [17], Amiri et al. [19] introduced a virtual simulator that performs on a grid for data distribution tasks. That simulation entity is based on two phases: preprocessing and post-processing.

An et al. [20] have introduced a parallel architecture for network discovery system, which focuses on the utilization of new-generation CPUs and GPUs. They have used a GPU-CPU buffer manager and a GPU-based view creator. They have implemented an ensemble algorithm to deal with the frequent pattern mapping algorithm.

Wang et al. [21], [22] have described an outstanding consequence about multicore clusters. They performed on different twin CPU multicore systems with 4 to 8 cores. They uncovered a good speedup on a large space production. A remarkable finding about the performance of hashing algorithm is presented by Engel et al. [23]. Even after

having these methods to distribute data-mining equipment in a cluster or grid, no better technique was made to effectively use them on any single machine.

Li et al. [57] has analyzed a similar concept on NERSC's Perlmutter, a State-of-the-art open science HPC system with both CPU-only and GPU-accelerated nodes. They have found a significant result. The supercomputer which they have used is world's top 8 supercomputer, and they have found that about 50% of GPU-enabled jobs uses up to 25% of the GPU memory, and the capacity was not fully utilized in some ways for all jobs.

B. PERFORMANCE PREDICTION

Several researchers [24], [25], [26], [27], have demonstrated a simulation-based performance prediction method, which criticizes the microarchitecture elements of CPUs. The proposed work uses a DNN-based model [28], a state-of-the-art algorithm for existing machine learning methods. Various features are available for the DNN model as buffer size for read-write operation and pipeline stages. In this paper, we are not using these micro-architectural features except the CPU specifications provided by manufacturers. Some researchers [29], [30], [31], [32], [33], [34], have developed a mechanistic model to predict the workload on CPUs. An empirical model [35], [36], [37] has been used with machine learning models [38], [39] to improve the performance of multi-core systems. Zheng et al. [40] have used a dynamic coupling-based performance counter method for performance estimation.

All the authors, as mentioned earlier, have used micro-architecture-based features which would help to make performance predictions accurately. Performance predictions and selection of the CPU component is a well-studied area. Piccart et al. [41] have used the data transposition technique for machine similarity.

C. WORKLOAD CHARACTERIZATION

Based on the workload description, Phansalkar et al. have designed excellent work [10], [42], [43]. However, they do not discuss the performance and scaling of the CPU, so their obtained results are not consistent.

In this paper, one of our objectives is performance prediction. Shelepov et al. [44], [45] use static workload features, which include cache size, spatial locality, frequency, etc., to predict performance.

They have also indicated the speedup for such systems, including different processors. Apart from this, many researchers have done work based on cloud scheduling [46], [47] to study workload characterization. A feature-based study done by Hoste et al. [48], [49] examines such features independent of micro-architecture to categorize the workload. They have regarded the locality and ILP metrics to indicate performance. The locality includes LRU stack models, Independent reference models, density functions (temporal and spatial), set models, and memory reuse distance models [50], [51], [52], [53].

TABLE 2. Details of used symbols.

Symbol	Meaning
M_{span}	Make span
$L_{balance}$	Load balance
$E_{consume}$	Energy consumption
L_{ik}	Load on processor p
W_i	Weighted sum of load resources
I_1, I_2	Individuals to share the information.
M_{info}	Mutual Information
H_I	Entropy
D_I	Normal distribution
$Norm()$	Normalized Information
I_{contri}	Contribution of I candidate
f_{cap}	Feature capabilities
I_{select}	Selected feature
SD	Standard Deviation
μ	Mean
$E(I_{contri})$	Evaluation of contribution of I candidate
PBM	Predicted Base Model

D. RESOURCE PREDICTION

Most of the researchers have done their work in the cloud computing environment. There is various little work done in the field of High-Performance Computing. Here we are trying to conclude those works necessary to process the resource prediction graphs in our model. The Resource prediction methods are classified into two fields in scientific computing, i.e. Processor Load prediction and Instance prediction based on running applications [17]. This paper uses both categories. The dataset preprocessing division and model prediction phase will forecast the load of HPC systems. Similar work has been done by Nguyen et al. [18]. He has proposed an adaptive workload scheduling method for automatic resource management. They used GC-traces as their dataset to evaluate the experimental model.

Similarly, Amiri and Mohammad-Khanli [19] have proposed a hybrid method by using reinforcement learning. The significant contribution of their research work is that the model can predict the future demand of complex scientific computing resources based on current workload. A fuzzy-based self-adaptive model has been proposed by Zheng et al. [40] to predict the demand for used resources.

Based on the above-mentioned related works, it is a very tedious task to decompose the simulation environment with used applications and their independent subtasks. However, the existing models fail only because of a single common error i.e. they won't consider the required characteristics of the simulation applications for execution in the physical environmental setup and hence, cannot achieve higher accuracy.

III. NOTATION USED, ASSUMPTIONS AND DATASET DESCRIPTION

A. NOTATION USED

All the used symbols in this paper are listed in Table-2, which will be required to understand the problem formulation and other used descriptions. The equations have also been described in detail throughout the respective sections based on their use.

TABLE 3. Assumptions for parallel computing architectures.

Features	Assumptions in parallel computing
Priori Process coordination	Yes
Homogeneity on Network, Host and processes	Yes
Network speed	Fast
Network Reliability	High
Host speed	Fast
Host Reliability	High

B. ASSUMPTIONS

All the assumptions that will be needed for this paper, are summarized in Table-3 and described below in detail.

- **Priority process coordination:**All processes have a priority level assigned to them that determines their execution order. The scheduling algorithm ensures tht higher priority processes are executed first. The processes with the same priority are executed in FCFS manner.
- **Homogeneity on Network, Host and Processes:** All nodes in the network have the same hardware configuration and operating system. All processes running on each node have the same memory and CPU requirements. The network is configured to have identical bandwidth, latency and packet loss rates for all the nodes.
- **Network Speed:** The network has a high bandwidth and low latency to ensure fast data transfer between nodes. The network is designed to handle a large volume of data traffic without any congestion or packet loss. The network supports multiple protocols and data transfer modes to optimize performance based on the type of workload.
- **Network Reliability:** The networks is designed to be fault-tolerant and resilient to network failures. Redundant paths are available for data transfer to ensure continuous operation in case of network failure. The network has monitoring and diagnostic tools to detect and resolve issues in real-time.
- **Host Speed:** Each node has a fast processor and ample memory to handle the workload assigned to it. The operating system is optimized for parallel processing and can efficiently schedule tasks across multiple cores. Each node has access to high-speed storage to enable quick read/write operation.
- **Host Reliability:** Each node has redundant components such as power supplies, fans, and network interface cards to minimize downtime due to hardware failures. The operating system and applications running on each node are monitored for stability and performance. Automated backup and restore mechanisms are in place to ensure data integrity in case of a host failure.

C. DATASET DESCRIPTION

The parameters of the models were modified to effect distinct simulation applications. After that, the simulation applications were deployed in a cloud environment. The simulation datasets were developed by gathering information regarding simulation applications. The Phold datasets includes 10050 samples, and the SOS datasets possesses 8010 samples. The collected components are shown in Table-4. The target of the feature is the number of CPU cores assigned for the simulation application with the minimum execution time.

TABLE 4. Illustration of the Features.

Features	Description
CPU usage	%age use of cpu during execution
File	%age use of file system during execution
Memory	%age use of memory in bytes during execution
Network receive	%age use in bytes of data during execution
Network send	%age use of bytes of sent data during execution
Execution time	execution time
Pre-allocation resource	allocation of resources
Network delay	delay
LA	look ahead buffer

IV. PROBLEM FORMULATION

The problem is to schedule the workflow and predict the sequence of execution of tasks. So that the required long-term resources can be easily identified. For doing this, the objective of our work is to reduce the make-span, load-balance and energy consumption according to the maximum function described in eq. (1).

$$max_{Fn} = w_0 \times \frac{1}{M_{span}} + w_1 \times \frac{1}{L_{balance}} + w_2 \times \frac{1}{E_{consume}} \quad (1)$$

As in earlier section-I, we have already introduced our objectives, which are conflicting in nature. So, we have used the weighted sum method introduced by [18] and [29]. These objectives are bound to sum up with the weight by the following relation described in the below conditional equation.

$$\sum_{i=1}^{k} w_i = \begin{cases} 1 & \text{if } k \leq 3 \\ 0 & \text{Otherwise} \end{cases}$$

Further, let L_{ik} will be the load on processor p, such that:

$$L_{ik} = \begin{cases} 1 & \text{if task } t_i \text{ is executed by processor } p_k \\ 0 & \text{Otherwise} \end{cases}$$

By using w_i and L_{ik}, we get:

$$\sum_{j=1}^{n} L_{jk} = \begin{cases} 1 & \{\forall\, j \mid 1 \leq j \leq n\} \\ 0 & \text{Otherwise} \end{cases}$$

$$\sum_{j=1}^{3} w_j = 1, \{\forall\, j \mid 0 < j \leq 3\}$$

where, L_{ik} is the load and w_i is described as the weight of the loaded resources on processor p_k. The assigned tasks will also be passed through our proposed framework to determine the respective execution time with its consumed voltage and frequency. So our job is to predict w_i with respect to p_k according to eq. (1).

V. PROPOSED ALGORITHM

In this section, we will first briefly introduce the ensemble algorithm and then we will discuss the different phases of our proposed algorithm in detail. We will also describe the evaluation scheme and metrics to show the effectiveness of our proposed algorithm.

A. OVERVIEW OF ENSEMBLE ALGORITHM

In figure-1, we have designed a flow diagram of our proposed framework. The proposed algorithm is used to train multiple independent models by using bagging technique in this floe diagram. We have proposed Feature Based Compatibility Prediction Algorithm (FBCA) for feature selection. We have also used this algorithm for the selection of optimal feature among all the training sets. To improve its quality and effectiveness, we have proposed one more algorithm entitled, ARRE in terms of use. As shown in figure-1, the long-term resource prediction will be calculated by using FBCA and ARRE algorithms. The components of the flow diagram are described below.

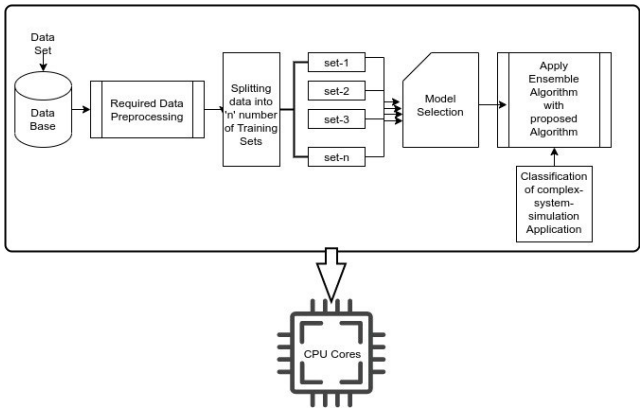

FIGURE 1. Flow diagram of dataset processing and passing through the CPU cores.

- **Dataset**: A huge amount of data is required to predict required long-term resources. So we have selected the application prototype to generate the simulation-based datasets. These generated datasets are best suited based on their characterization of features. We have built different samples from the obtained datasets by applying different configuration parameters such as maximum main memory, minimum main memory, maximum cache memory, minimum cache memory, processing frequency, etc. [8]. We have selected a standard FP benchmark [6] for the deployment of our generated datasets.

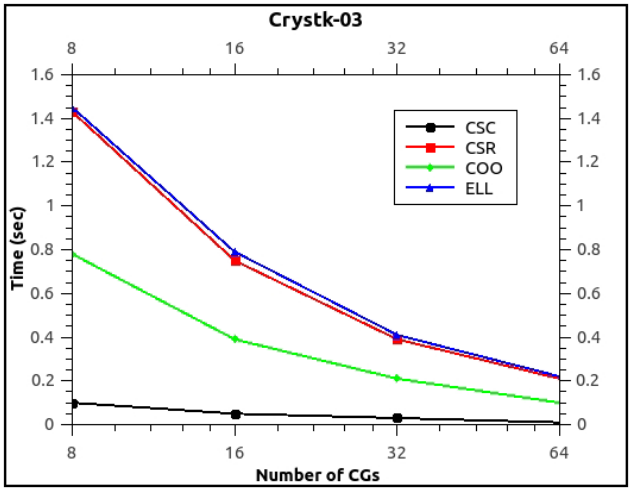

FIGURE 2. Feature distribution based on number of SPECs (CGs) and Time domain (loading time).

- **Data preprocessing**: To improve the quality of any dataset, preprocessing is a necessary phase. We have done our experiments in two parts: first part will cover the data cleaning and the second part will cover transformation. The data cleaning part is required to identify the outliers. The feature distribution has shown in the form of a line-graph, shown in the figure-2.
The figure is made up by excluding the outliers with respect to the baseline. The outliers are the instances plotted outside of the lower and upper bound from the base-line. The transformation part covers a replacement of instances. Suppose that, if the value of any outliers shows a large difference from its neighbor instance, then it will replace the such value with its mean value. If the outliers show a reasonable difference, then the transformation part will not process. After the replacement of the value of the instance form dataset, the transformation part also does a format conversion to process the proposed Algorithm. After that, we have used a normalized form of dataset. This normalization will done by the Z-score technique.
The Z-score technique is feasible to normalize SPEC (Standard Performance Evaluation Corporation) dataset because it transforms the original data to a standard normal distribution, which has a mean of zero and a standard deviation of one. This transformation makes it easier to compare and analyze data across different experiments and benchmark. The Z-score technique helps skewness and kurtosis of the distribution by centering the data around the mean and scaling it by the std. deviation. The Skewness measures the asymmetry of the distribution, while kurtosis measures the degree of peak or flatness of the distribution. When using the Z-Score technique, data points that are farther away from the mean will have a higher absolute Z-score, indicating that they are outliers or extreme values.

By standardizing the data, the effort of outliers is reduced, and the distribution becomes more symmetrical with lower skewness and kurtosis values.

We have tried min-max scaling, decimal scaling for normalizing data. The main difference between these techniques and Z-score technique for normalizing SPEC dataset lies in their scaling capability of the data. The min-max scaling scales the data to a specific range(between 0 to 1 or -1 to 1) by subtracting the minimum value and dividing by the range. This method can be affected by outliers and can result in loss of information.

The Decimal scaling involves scaling the data by dividing by a power of 10, such as 10 or 100, to move the decimal point to the left. This method can be simple and efficient but does not take into account the distribution of the data. On the other-hand, the Z-score technique scales the data by subtracting the mean and dividing by the standard deviation, resulting in a standard distribution with a mean of 0 and std-deviation of 1. This techniques takes into account the distribution of the data, is not affected by outliers, and preserves the information in the original data.

In the selected dataset, the Z-score technique is a better normalization method because it is more robust to outliers and can better handle variations in the data. This important in performance evaluation where benchmarks can vary widely in terms of their characteristics and performance results. Additionally, the Z-score technique is widely used and well understood in the field of statistics and machine learning, making it a more reliable and trusted method for normalizing SPEC dataset.

- **Feature selection**: Feature selection is one of the best phases in machine learning techniques because by using this method one can improve the prediction of accuracy and time complexity. Both of these (time and accuracy) constraints play a vital role in problem-solving and Algorithms. Therefore, feature selection also becomes a crucial task to select the best set of features. The main advantage of feature selection is that it reduces the redundancies of selected features and improves the performance of simulation application algorithms. Webar et al. [35] have found that most of the feature selection algorithms are based on two different features: the first one is maximizing the relevance instance value and the second one is to minimizing the redundancies. In this paper, we have proposed a FBCA algorithm shown in Algorithm-1, to create an impact over existing feature selection.

Suppose M_{info} is the mutual information shared between two individuals I_1 and I_2, then the measurement of dependence between I_1 and I_2 will be written as $M_{info}(I_1, I_2)$. The entropy $H(I_1)$ is defined as the dependence disorder or randomness or uncertainty of the mutual information. The uncertainty measurement will be done on 36 instances, represented as I[36]. The relation between $M_{info}(I_1, I_2)$ and

Algorithm 1 Feature Based Capability Prediction Algorithm (FBCA)

Input: *Feature_Set* : F, *class_Set* : D, *No._of _features* : n, *Selected_features* : x
Output: *Selected_Set_of _features* : F_s
Initialisation: $t = 1$
1: **for** i=1 to n **do**
  Compute weigt(F_i, D)
2: **end for**
3: Select feature $F_m|max\_weight(F_i, D)$
4: $F_s = F_s\ U\ F_m$ //
$F_s = F_{s1}\ U\ F_{s2}\ U\ F_{s3}\ U\ \ldots\ U\ F_{sn}$
5: **for** x=1 to m **do**
  if $(x > t)$
    remove F_m from F
  end if
6: **end for**
7: **for** i=1 to Len(F) **do**
  Compute $F(x_i)$
8: Select the feature set $F_x|max\_weight(F_x, D)$
  $F_s = F_s\ U\ F_x$
  $t = t + 1$
9: **end for**
10: **return** F_s

$H(I_1)$ is represented by the below-mentioned equation (eq. 2):

$$
\begin{aligned}
M_{info} &= \sum_{I_1 \in I} \sum_{I_2 \in I} D(I_1, I_2) log(\frac{D(I_1, I_2)}{D(I_1)D(I_2)}) \\
H(I_1) &= \sum_{I_1 \in I} D(I_1) log(I_1) \\
H(I_2) &= \sum_{I_2 \in I} D(I_2) log(I_2)
\end{aligned} \tag{2}
$$

$D(I_1)$ and $D(I_2)$ are defined as the normal distribution of I_1 and I_2. $D(I_1, I_2)$ is defined as joint distribution of I_1 and I_2.

We have chosen normal and joint distribution for the above equation (eq. 2) because the mutual information will be the intermediate part of the information from both individuals. So, the normalized mutual information will be written as eq. 3:

$$
Norm(M_{info}(I_1, I_2)) = 2 \times \frac{M_{info}(I_1, I_2)}{H(I_1) + H(I_2)} \tag{3}
$$

Since the information depends upon both individuals I_1 and I_2. So, we have multiplied our normalized mutual information with 2. Also the it is notable that the distribution is joint in nature with respect to I_1 and I_2. So, we have used their entropy in an additive manner.

To calculate the feature capabilities of the contribution of individual *Is* we have assumed that the Contribution of I_1 candidate feature is $I_{contri}|(I_1, I_2) \in C$, where C is specific feature class for I_{contri}. So, the feature capabilities (f_{cap}) is

defined as eq. 4:

$$f_{cap}(I_{contri}, C) = Norm(M_{info}, C) + Const \times (1 - (\frac{H(I_{contri})}{log\ k})) \quad (4)$$

where $Const$ is any constant $0 \leq Const \leq k$ defined as the weight between the distribution of mutual information $D(I_{contri}, C)$, and k is the total number of contributed candidate features (I_{contri}) of (I_1, I_2) categories.

Based on the above assumptions, the redundancies between selected feature (I_{select}) and contributed feature (I_{conti}) can be defined as eq. 5:

$$M_{info}(I_{contri}, I_{select}) = \begin{cases} \text{Overall} & \forall(\ I_{contri} \leq \text{redundancy} \leq I_{select}) \\ \text{Ignored} & (if\ (I_{contri} - I_{select}) = 0) \end{cases} \quad (5)$$

Therefore, we have used standard deviation (SD) as $Const.$, defined in $f_{cap}(I_{contri}, I_{select})$ formula. The standard deviation (SD) is a measure of the amount of variation or dispersion of a set of values. A low SD indicates the expected value, since the value tends to be close to the mean, whereas a high SD indicates that the values are spread out over a wider range.

A detailed definition of SD is shown below in eq. 6:

Let μ be the expected value (the average) of distribution $D(I_1, I_2)$ with density of selected features I_{select}. Then,

$$\mu \equiv \frac{1}{\mid I_{select} \mid} \sum_{I_{select}(I_1,I_2) \in I_{contri}} Norm(M_{info}(I_{contri}, I_{select})) \quad (6)$$

The standard deviation (SD) denoted as σ of I_{select} is defined as, shown in the equation at the bottom of the page.

This is because the standard deviation is the square root of the variance of I_{contri} & I_{select} with respect to I_1 & I_2. Based on the above formula if the standard deviation σ is low then the redundancy between I_{contri} and I_{select} is closer to mean μ. To evaluate its contribution we have expressed $E(I_{contri})$ in the following terms:

$$E(I_{contri}) = f_{cap}(I_{contri,C}) - const. \times (1 - \sigma) \sum_{I_{select}(I_1,I_2) \in I_{contri}} Norm(M_{info}(I_{contri}, I_{select})) \quad (7)$$

The $(1 - \sigma) \sum_{I_{select}(I_1,I_2) \in I_{contri}} Norm(M_{info}(I_{contri}, I_{select}))$ is defined as the representation of both overall and individual feature selection between I_{contri} and I_{select}.

The proposed algorithm 1 is divided into five major steps. The steps are described as follows:

1) Compute the feature capability of all features available in feature-set (*i.e.* $Feature_{Set} : F$).
2) Select the feature, which has the highest feature capability and make the time-span ($t = 1$).
3) Based on feature selection criteria, rank them into their corresponding computed scores and evaluate them.
4) Merge the feature having the highest capability score to the feature set and increase the time span (*i.e.* $t = t+1$)
5) Repeat the steps from 3 to 5 until the selected feature has highest score (*i.e.* $x > t$)

- **Training and Model Selection**: We have selected supervised learning to predict the number of required CPU cores. We have selected supervised learning because the problem discussed in section IV is a classification-based problem and the mapping of input to output level is continuous. We used K-nearest neighbor (KNN), Decision Tree (DT) and Support Vector Machine (SVM) with different parameters to generate the base model for CPU requirement prediction and simulation of applications. We have tried several Algorithms including Naive Bayes, Gradient Boosting machines, KNN, DT and SVM and found that the SVM, KNN and DT generate the base model more accurately. A comparison of Different algorithms has show in Table-5. The reason behind choosing the SVM is, SVM is known to be effective in handling non-linear relationship between the features and the target variables. It is also important in performance evaluation where, the relationship between features and performance results may not be linear. The SPEC dataset has the same favorable features. KNN is best useful algorithm in handling noisy data. The SPEC dataset has some noise due to variations in hardware configurations, operating systems, and other factors (Min-Memory, Max-Memory, Min-Clock, Min-Cache, Min-Frequency etc.) Similarly the decision tree is well-suited for all datasets with a mixture of continuous and categorical variables, which is often the cause in performance evaluation where benchmarks have different types of features. It can also handle missing values, in our dataset missing values occurs due to measurement errors. The Naive Bayes and Gradient Boosting Algorithms are supportive only when all the features are independent, which is not true in our case.

  A sequential and parallel execution of our selected dataset (SPEC) has shown in figure-3, 4, 5 and figure-6.

  The total dataset size is 3 terabytes (TB). The SPEC CPU benchmark uses a base reference dataset size

$$\sigma \equiv \sqrt{\frac{1}{\mid I_{select} \mid} \sum_{I_{select}(I_1,I_2) \in I_{contri}} ((Norm(M_{info}(I_{contri}, I_{select}))) - \mu)^2}$$

TABLE 5. Comparison of different normalization algorithms, based on accuracy, precision, recall and F1-Score for SPEC dataset.

Classifier	Accuracy	Precision	Recall	F1-Score
Naive Bayes	0.82	0.84	0.86	0.85
Gradient Boosting	0.85	0.84	0.83	0.82
SVM	0.92	0.92	0.92	0.91
KNN	0.92	0.91	0.94	0.92
DT	0.88	0.87	0.89	0.88

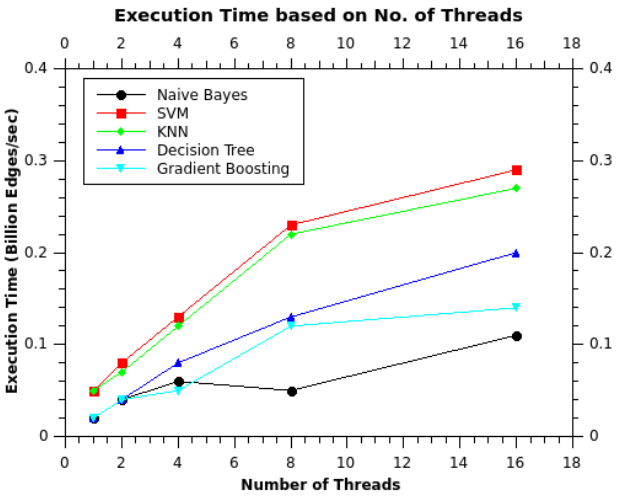

FIGURE 3. Uniform execution of algorithms (SVM, KNN, DT, GBT, and Naive Bayes).

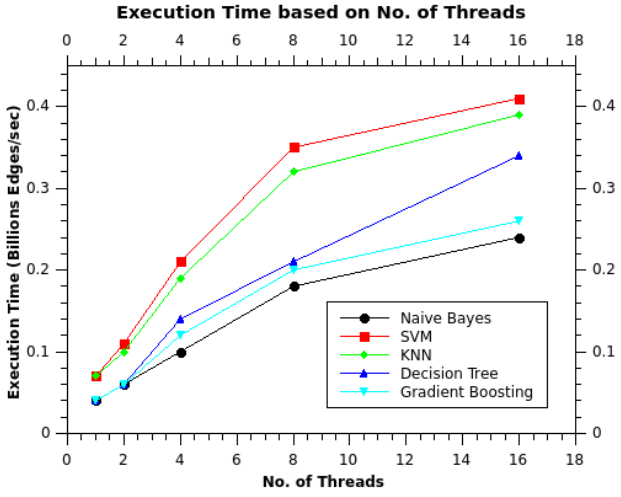

FIGURE 4. Random execution of algorithms (SVM, KNN, DT, GBT, and Naive Bayes).

of 1000 MB. In this case the dataset scale is as follows:

$$Dataset\ Scale = \frac{Total\ Dataset\ Size}{Base\ Reference\ Dataset\ Size}$$

$$\text{So, } Dataset\ Scale = \frac{3\ TB}{1000\ MB} = 3,000$$

So, the dataset scale is 3,000. This means that the working dataset used in this benchmark is 3,000 times larger than the base reference dataset.

We have divided the dataset into two categories: training and test datasets for the better use of simulation

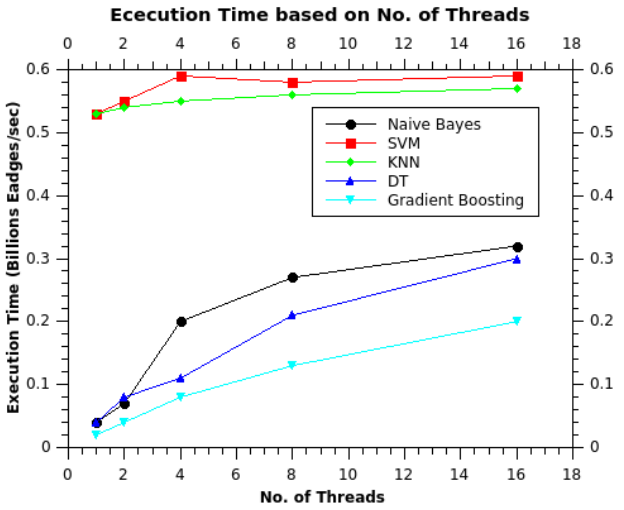

FIGURE 5. Parallel uniform execution of algorithms (SVM, KNN, DT, GBT, and Naive Bayes).

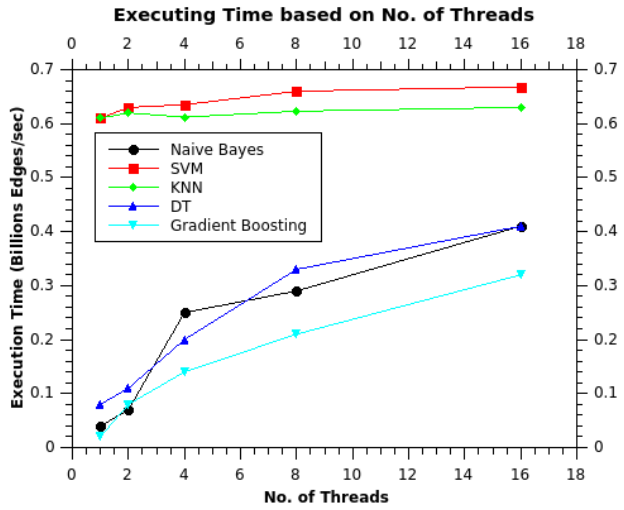

FIGURE 6. Parallel random execution of algorithms (SVM, KNN, DT, GBT, and Naive Bayes).

applications. We have also used a similar approach to train and test our simulation mode as any ML system does.

For model selection, we have used the ensemble model selection techniques, because, as we know, a single ML model has several limitations in meeting the computation requirements of a complex system, and ensemble model uses a combination of multiple base models to achieve better results. These models have a better generalization with respect to local optima as compared to any single ML model.

For the basic understanding, ensemble models are defined as an integration of different base models. In such base models whenever the accuracy increases the diversity will also increase. Based on its accuracy and diversity, its performance will also increase. So, for our simulation, we have to increase the disturbance of

the sample data set to train our base model. This will increase the existing difference between the base and selected models. It will also improve its generalization performance.

Notably, none of any base models has that much of a positive impact on ensemble techniques based on performance. So, we have proposed Algorithm 2, which will be able to choose the correct set of base models as per the user's requirement. Since it will be able to choose the required model as per the user's need. It can also reduce the execution time by discarding the timestamp selection of weak models. The proposed Algorithm 2 is shown below:

Algorithm 2 Accuracy and Relative Runtime Error (ARRE)

Input: *Base Model Sets* $\{BM_1, BM_2, BM_3, \ldots, BM_n\}$, *Dataset*$(D_s)$

Output: *Predicated Base model sets*$\{PBM_1, PBM_2, \ldots, PBM_k\}$

Initialization: $t = 1$

1: **for** i=1 to n **do**
   Compute weigt(F_i, D)
2: **end for**
3: Select feature $F_m | max\_weight(F_i, D)$
4: $F_s = F_s \ U \ F_m$
   // $F_s = F_{s1} \ U \ F_{s2} \ U \ F_{s3} \ U \ \ldots \ U \ F_{sn}$
5: **for** x=1 to m **do**
   if $(x > t)$
   remove F_m from F
   end if
6: **end for**
7: **for** i=1 to Len(F) **do**
   Compute $F(x_i)$
8: Select the feature set $F_x | max\_weight(F_x, D)$
   $F_s = F_s \ U \ F_x$
   $t = t + 1$
9: **end for**
10: **return** F_s

Algorithm Explanation: The Proposed ARRE Algorithm-2 is processed into 4 steps. The corresponding steps are described as follows:

1) After taking a set of base models as input, the algorithm computes each base model's Accuracy and Relative Error.
2) The obtained accuracy and relative error values will be arranged into ascending order (the algorithm will perform a sorting operation).
3) Select all the based models having the same or higher accuracy and relative error.
4) Apply the ensemble technique to select the best-combined base models.
5) Finally, select the best base model set based on their maximum accuracy and minimum relative error.

Once the predicted base models has obtained then the ensemble technique has applied on them. The following formula (eq. 8) is used to apply ensemble technique:

$$Ensemble\_I_{predict}\{PBM_1 \ U \ PBM_2 \ U \ \ldots \ U \ PBM_k\} = I_{select}(\Sigma_{j=1}^{t} PBM_j\{PBM_1 \ U \ PBM_2 \ U \ \ldots \ U \ PBM_t\}) \tag{8}$$

where $I_{predict}\{PBM_1 \ U \ PBM_2 \ U \ \ldots \ U \ PBM_k\}$ is the information predication among predictive base models sets, is the information class described earlier.

B. PERFORMANCE EVALUATION OF PROPOSED ALGORITHMS

The performance of any model depends on the following characteristics: Accuracy, RMSE, F1-Score and Relative Error. So, to evaluate the performance measurement of our proposed Algorithm, we have also used the above-said metrics, which are defined as follows:

$$\begin{aligned} Accuracy &= \frac{1}{n}\sum_{i=1}^{n}[I_{predict}[i] = I_{select}[i]] \\ RMSE &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}[I_{select} - I_{predict}]^2} \\ Pi &= \frac{TP_i}{TP_i + FP_i}, \\ Ri &= \frac{TP_i}{TP_i + FN_i}, \ and \\ F1 - Score &= \frac{1}{C}\sum_{i=1}^{n}\frac{2PiRi}{Pi + Ri} \end{aligned} \tag{9}$$

$$RelativeError = \lambda * Accuracy + (1 - \lambda) * (1 - \frac{\sum_{i=1}^{n}[I_{select} - I_{predict}]^2}{\sum_{i=1}^{n}[I_{select} - \bar{I}_{predict}]^2}) \tag{10}$$

where, TP_i, FP_i, FN_i are true positive, false positive, and false negative scores for the $I_{predict}$ and I_{select} class. Hence, the selection of a number of CPU core is referred to I_{select} and $I_{predict}$. n is the total number of samples, and C is the total number of classes. λ is the weight to be selected according to the user's performance requirement and accuracy. The Relative Error is only the indicator for comprehensive computation of accuracy and Deviation of TP_i and $I_{predict}$.

VI. EXPERIMENTAL SETUP AND SIMULATION

We used Phold [27], [39] and SoS (Social opinion systems) in our experiments. Both of these applications are CPU-Intensive and supported by resource prediction algorithms. To find better performance, we have done different studies on the configuration of these applications. The detailed configuration parameters are shown in the table-3.

TABLE 6. Benchmark with the number of instances used for simulation.

Used Benchmark	No. of Instances used	Remarks
Phold	(50,400)	No. of objects participated in Simulation
	(10,100)	Objects/Events generated from Simulation
	(1000)	Total running time of Simulation/Experiments
	Preserved	Time Distribution based on resources
	(0.1,1)	Look Ahead Carry Buffer
Social Opinion System	[100, 1000]	No. of Profiles
	(5,20)	No. of Cities Selected
	(10)	No. of Infrastructure
	(5%,20%)	Probability of IP Networks
	(1000)	Total running time of Simulation/Experiments
	Preserved	Time Distribution based on resources
	(0.1,1)	Look Ahead Carry Buffer

A. PHOLD

In the Parallel and Discrete Simulation Environment (PEDS) [13], the Phold is the most powerful and effective performance evaluation benchmark. Many LPs (Linear programming simulation object models) are connected with Phold. All events pass through the event simulator object state. These transferred events are classified into three major categorical objects (i.e. Local, Autonomous, and Simulation object/Event). The detailed structure of Phold is shown in the figure-7.

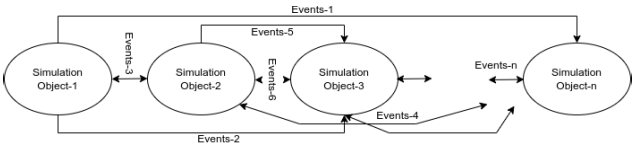

FIGURE 7. Phold structure with *n* simulation objects.

In figure-7, the simulation of Phold benchmark has shown. A single simulation object contains the following set of components and processed in different set of steps

- Initialization: At the start of the simulation, *n* simulation objects are created. Each objects have its initial state, attributes, and defined behavior.
- Event Generation: Each simulation object can generate events at a specific time based on their pre-defined conditions. These events has shown as event-1, event-2, ..., event-n in the figure-7.
- Event Queue: All the events generated are placed into an event queue in FCFS order. All the events are further processed by simulation objects.
- Event Execution: Event execution and an execution clock keeps track of the current executed event and simulation.

B. SOS

SOS is relatively typical and complex in the parallel & discrete simulation environment because it can map the interrelationships of remarks mentioned in the table-6. In the table the remarks of each simulation object has explained, the details are as follows:

- **No. of objects participated in Simulation (50,400):** This remark indicates that the simulation involved a substantial number of objects, with a total count of 50,400. Each object likely represents an entity or element within the simulated environment, showcasing the scalability and complexity of the simulation.
- **Objects/Events generated from Simulation (10,100):** The simulation generated a notable quantity of objects or events, totaling 10,100. This metric reflects the dynamic nature of the simulation, where various objects interacted or events unfolded, contributing to the overall complexity and richness of the simulated environment.
- **Total running time of Simulation/Experiments Phold (1000):** The running time for the Phold simulation or experiments reached a significant duration, clocking in at 1000 units. This indicates the duration for which the simulation was executed, providing insights into the time scale over which the simulated events and processes unfolded.
- **Preserved Time Distribution based on resources (0.1,1):** This remark suggests that the simulation considered a time distribution based on available resources, with a range of 0.1 to 1. It implies that the simulation took into account varying resource availability, possibly simulating scenarios where resources fluctuated within this specified range.
- **Look Ahead Carry Buffer [100, 1000]:** The inclusion of a look-ahead carry buffer with a range of 100 to 1000 implies that the simulation implemented a mechanism to anticipate future events or conditions. This buffer range could influence decision-making within the simulation, allowing for a degree of foresight in the simulated environment.
- **No. of Profiles (5,20):** The simulation involved a range of profiles, with the number falling between 5 and 20. These profiles likely represent distinct configurations or characteristics within the simulated entities, contributing to the diversity and variability of the simulated population.
- **No. of Cities Selected (10):** In this simulation, 10 cities were selected, suggesting a spatial or geographical component to the simulated scenario. The choice of cities as entities may be relevant to simulations involving urban planning, resource distribution, or other city-centric scenarios.
- **No. of Infrastructure Social Opinion System:** The simulation incorporated an infrastructure social opinion system, indicating that social opinions within the simulated environment were influenced by an underlying

infrastructure. This feature adds a layer of complexity, as social dynamics are intertwined with the state of the infrastructure.

- **Probability of IP Networks (5%,20%):** The probability range of 5% to 20% for IP networks suggests that the simulation considered varying probabilities for the existence or utilization of IP networks. This could reflect scenarios where network connectivity plays a role in the overall dynamics of the simulated environment.
- **Total running time of Simulation/Experiments Preserved Time Distribution based on resources (1000):** Similar to the earlier mention, this remark reiterates the total running time of the simulation or experiments, emphasizing the importance of the 1000-unit duration. The preserved time distribution based on resources further underscores the consideration of resource dynamics throughout the simulation.
- **Look Ahead Carry Buffer (0.1,1):** This repetition emphasizes the utilization of a look-ahead carry buffer with a range of 0.1 to 1, highlighting its significance in influencing the simulation's decision-making processes.

In summary, these remarks collectively describe the intricacies of the simulation, including the scale of entities involved, time considerations, resource dynamics, and the incorporation of specific features like profiles, cities, infrastructure, and social opinion systems.

It becomes more complex when the infrastructure is not available or destroyed. the infrastructure includes electric power supply, key resources of cities, etc. Based on these infrastructures, we can say that this can influence the person. The cities will also play a major role while making policies, based on infrastructure changes. A basic structure of SOS has shown in Figure-8.

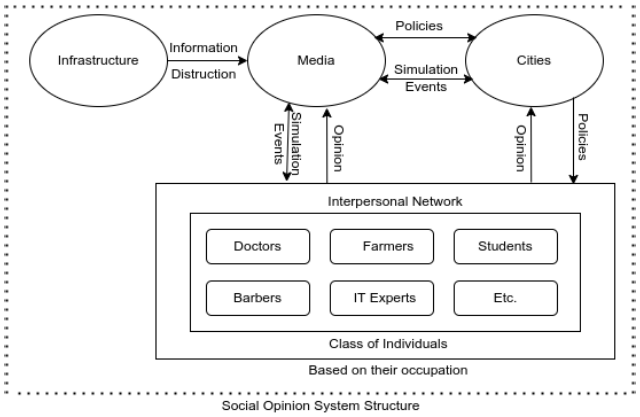

FIGURE 8. Social opinion system structure.

In the above figure-8, the infrastructure represents the physical and digital framework of a society, including communication network, Internet access and storage facilities. The media represents various channels and platforms through which information is disseminated to the public. Cities are influenced by polices made by authorities based on the feedback and opinion.

TABLE 7. Selected node configuration.

Nehalem CPU	Core CPU	Tesla GPU	SC10-EP	SC10-EX
Core Archi.	Intel Nehalem	Nvidia Fermi	Intel Core	Nvidia Tesla
Model No.	Xeon X5550	Tesla C2050	Xeon E5345	GeForce GTX275
Core Freq.	2.67 GHz	1.15 GHz	2.33 GHZ	1.40 GHz
No. Socket	2	1	2	1
No. Cores	4	14*2(a)	4	30
HW-threads	2	≈32	1	≈32
SIMD/SIMT	- (not used)	32	-	32
Total Cache	16 MB	2MB	8 MB	-
Main Memory	24 GB	3GB	32 GB	896MB
Memory Band	100 GB/s	128 GB/s	10.4 GB/s	127 GB/s
No. Transistors	1.4 Billion	3.0 Billion	1.1 Billion	1.4 Billion
Total Power	190 W	238 W	160 W	210 W

C. DATA SET DESCRIPTION

All the datasets are standard and taken from the official website of Phold-IBM.[2] We have adjusted the parameters from the datasets to generate the simulation application. The adjustment of datasets are already described in Algorithms (1 and 2), used benchmark for simulation in Table-6. The adjustments are followed by 10-step procedure which includes; Defining the event generation, configuring events, event queues for FCFS scheduling, event processing, setting of simulation time and termination criteria, collecting the results and analyzing the data recursively followed by parameter tuning.

Then we deployed them to our virtual HPC environment. The Phold & SOS contains 10050 and 5010 samples, respectively. Table-6 contains a detailed description of these used data sets' features. The selected features are based on their minimum execution time for all the selected CPU cores.

VII. RESULTS AND DISCUSSIONS

This section describes the results and discussions in detail. In this section, we have calculated our proposed Algorithm.

A. EXPERIMENTAL SETUP

We have used Eight nodes having 64 bits Operating System. Every node is selected from a selected set of CPU components discussed and published earlier by us [1], [7]. Table-7 shows a detailed used configured devices as the set of nodes for this experiment.

For this experiment, we have installed a docker on our machine. Docker is equipped with a core and a minimum of 2GB memory. A detailed configuration of node has already mentioned in table-7 The long-term running resource prediction has shown in figure-9. In this structure the datasets are first passes through a preprocessing methodology and then based on the requirements the train and test sets will be divided. Based on the best suitable dataset the model has selected and bind-up with the Ensemble Algorithm and Applied Simulation Application to find the best suitable results. Then it passes through the computation nodes having pre-defined cores (defined in section-I)

As per the structure, the LPs are responsible for distributing load among all cores. One notable thing here is that each LP partition has the same CPU configurations in terms of

[2]ibm.com/docs/en/zos/2.3.0?topic=initialization-page-data-set-sizes

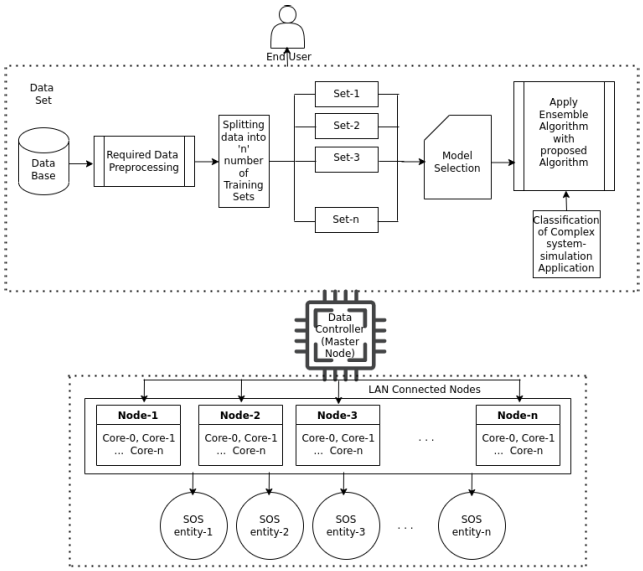

FIGURE 9. Long-term running resource prediction structure.

memory, frequency, cores, etc. The LP partition is based on Objective function (defined in proposed Algorithms-1,2), Decision Variables, resource allocation constraints, resource utilization constraints, model selection, forecasting and predicting the required results. In this architecture, the primary node is the controlling node, which collects the resource information. After collecting the resource information the primary node applies the algorithm to predict the computing resource.

B. EXPERIMENTAL RESULTS AND ANALYSIS

We have performed our analysis on three different aspects.1)On the basis of parameter selection, 2) On the basis of result validation, and 3) Based on Result comparison.

- **Parametric Experiments:** In this section we have discussed and analyze the performance of the model with different parameters. The proposed algorithm can be affected by multiple parameters, as discussed in the problem formulation section, i.e., feature selection dimension and ensemble of base models. The performance of the ensemble algorithm with selected parameters has calculated with 10-Fold cross-validation. The results has been shown in Figure-10. In our study, we chose to use k=10 in the 10-fold cross-validation for several reasons, and we did consider other values. The choice of k=10 is indeed a common and widely recommended practice in cross-validation for various datasets and machine learning tasks. Here's the rationale behind our decision:
  - **Common Practice:** k-fold cross-validation, where k=10, is a well-established and widely accepted standard in the machine learning communities.
  - **Bias-Variance Trade-off:** As we know that there is a bias-variance trade-off associated with the choice of k smaller values of k (k=5) can lead to higher variance in the estimated error rates, while larger values (k=9 or 10) tend to reduce bias. Our choice of k=10 was made with the aim of achieving a reasonable between bias and variance in the error rate estimates.
  - **Empirical Evidence:** Emprical studies and best practices have indicated that values like k=10 often yield stable and reliable estimates of test error rates without excessively high bias or variance.
  - **Computational Efficiency:** While k=10 provides robust estimates, it is also computationally efficient, especially when compared to larger values of k. This allows us to perform cross-validation efficiently, especially when dealing with SPEC datasets.

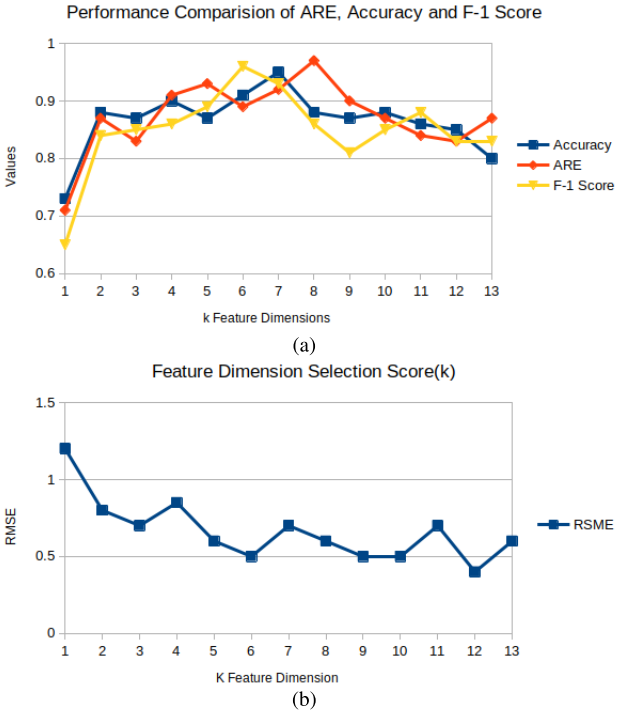

FIGURE 10. Performance comparison of intelligent ensemble algorithm with different feature dimensions.

In the Figure-10(a) the evaluation matrices have shown where the number of selected feature's dimensions increases, the respective accuracy, F-1 score and Absolute root mean error have increase once and then become stable within a specific range. After the range it decreases to some extent. When the number of folds i.e. k is equal to ≈ 9, the selected model shows an optimum performance. One thing is notable is that the low feature dimension representation is very difficult to represent because of its prediction task accuracy un-stability over a certain period of time. In the figure-10, it is observable that due to redundancy in features, the dimension gets over to a specific value, and the predicted performance has declined.

According to Figure-10(b) it has clearly observable that the RMSE starts declining when the k feature dimensions have increased. It is also observable that it remains stable for a certain period of time and then rises again. The RMSE will become lower within the range of 3 to 13.

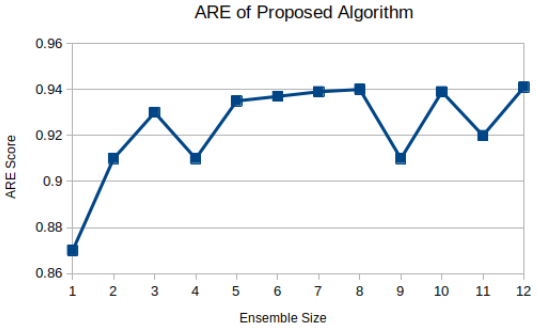

FIGURE 11. ARE of proposed algorithm.

Figure-11 shows different sizes of ensembles based on our proposed algorithms. The AREP method has been applied to choose the best base model for the experiment with the final predictor outcome. In this figure, it is also clearly shown that the ensemble is increasing for a range of periods and with this increase, the ARE is also increasing. When the ensemble size becomes 4 the ARE increases slowly up to 8. After an ensemble size of 8 the performance graph shows a slow increase in nature. This is just an indicator that the proposed algorithm becomes saturated after reaching a certain level. That is why we have used different base models for our experiments. The model descriptions are described in section III. Based on those base models, we have selected our intelligent ensemble algorithm to combine with our ARRE algorithm.

- **Experimental Result Verification:**
  In this section, we performed our experiments on the simulated virtual environment and validated our proposed algorithm. We have also compared our algorithm with existing algorithms (KNN, Traditional Bagging algorithm). One notable thing is that the KNN uses a grid-based searching technique to find its best optimal parameters, whereas the traditional Bagging algorithm uses a bagging technique to club its base models. Table - 8 contains a detailed comparison report of both KNN and traditional bagging techniques.
  From the above table-8, it is clear that our proposed model has a higher accuracy, and a lower root mean square error (RMSE). As we know that the RSME is the resultant of deviation between predicted and true values, so based on RSME, it is also clear that our proposed model is very good in terms of deviation. From the F1-score, the difference between performed activity between bagging and the proposed algorithm, the proposed algorithm shows a higher value $\approx$ 10%.

TABLE 8. Performance comparison of different selected model & proposed model.

Bench Mark	Model	Accuracy (%)	RSME	ARE(%)	F1-Score	Time
Phold	SVM	87.05	0.94	72.12	89.06	10.11
	DT	83.44	1.79	90.42	75.68	10.03
	KNN	89.63	1.23	90.37	92.83	08.93
	Bagging Ensemble	91.82	1.38	69.72	83.23	16.77
	proposed algorithm	94.11	0.06	93.54	94.68	11.31
SOS	SVM	84.71	1.02	86.31	90.32	10.34
	DT	87.69	0.93	89.64	82.86	12.08
	KNN	83.78	0.93	90.43	93.73	10.42
	Bagging Ensemble	89.28	1.02	79.32	86.84	59.23
	proposed algorithm	84.08	0.78	96.11	90.52	10.43

TABLE 9. Performance comparison between major resource prediction methods and proposed method.

Bench Mark	Model	Accuracy (%)	RSME	ARE(%)	F1-Score	Time
Phold	REAP	87.27	0.84	82.12	89.06	10.11
	BN	83.51	1.19	91.42	85.68	10.03
	LR	89.63	1.32	90.37	82.83	08.93
	RF	91.82	0.38	79.72	87.13	16.77
	FNN	91.82	0.38	90.72	81.29	16.77
	proposed algorithm	94.11	0.53	93.54	94.68	11.31
SOS	REAP	90.71	1.12	86.31	90.32	10.34
	BN	91.69	0.23	89.64	82.86	12.08
	LR	83.78	0.93	90.43	93.73	10.42
	RF	87.28	0.72	79.32	86.84	59.23
	FNN	89.28	0.46	79.32	86.84	59.23
	proposed algorithm	84.08	0.78	96.11	90.52	10.43

Based on the result in terms of time and F1-score, we can say that the Bagging algorithm performs lower then our proposed algorithm. The proposed algorithm is independent of the accuracy or true and false prediction. The compared method shows less training time, but our proposed algorithm's accuracy is 4% faster.
The proposed algorithm performs better because it chooses its base models wisely. The selection is based on Phold and SOS simulation services which reduce the influences of the poorly performed models.

- **Performance Comparison:** The table-9 shows a specific comparison between the proposed method and other major resource prediction methods like (the regressive ensemble approach for predicting resource (REAP[23]), Bayesian Approach (BN[24]), cost-aware auto-scaling approach (LR[25]), the advanced model for efficient workload prediction (RF[26]), and fuzzy neural network (FNN[20]).
  For all our experiments we have used Root mean square error and absolute root error at ($\lambda = 0.5$)%. We have also used F1-score to evaluate the comparative prediction with the indices parameter. During our experiments,

we have observed that the parameters of the comparative experiments needs to be discretized. A clear difference can be seen in Table-9.
In this table, it is also clear that our proposed algorithm is significantly better than other traditional models. Other methods like REAP can achieve good accuracy, but it is not ideal with other performance evaluation indices. If we see at BN and LR they take less amount of time but they also have a lower accuracy. The FNN requires more training time then the RMSE and proposed method. So, overall we can say that our proposed method can achieve high accuracy, low training time and a better user requirements to improve the QoS.

C. BENCHMARK SUBSETS VALIDATION

In this section, we will be validating all the created simulation entities. By doing this one can easily identify which set of simulation entities will be suitable to represent in modern CPUs and SPEC benchmarks. We used inter-process communication [1] and L1-cache memory-based miss ratio technique [7] for all the use benchmarks shown in Table-9. At last, we have compared it with other existing related works [23], [38], [39], [40], [54], [55].

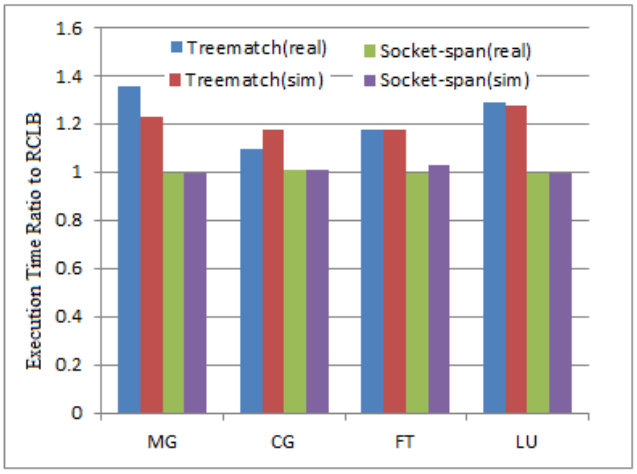

FIGURE 12. Evaluation of simulation entities used to check representation of modern CPUs and SPEC benchmarks.

- **Computing IPC:** Using the subset based on overall program characteristics, we calculated the average IPC of the entire suite for two different micro-architectures configurations with issue widths of 8 and 16. Figure-12 & 13 shows the average IPC of the entire benchmark suite calculated using the program subset and every program in the benchmark suite. We obtained the IPC on 8-way and 16-way issue widths for every program in the SPEC CPU2000 benchmarks from Huang et al. [54]. The configurations in brief are: 8-way machine (32KB 2 way L1 I/D cache, 1M 4-way L2, Functional Units 4 I-ALU, 2 I-MUL/DIV, 2 FP-ALU, 1 FP-MUL/DIV) and 16-way machine(64 KB 2-way L1 I/D, 2M 8-way L2, Functional Units 16 I-ALU, 8 I-MUL/DIV, 8 FP-ALU, 4 FP-MUL/DIV). The rest of the details about branch predictors and different penalties in cycles can be found in [28]. From Table-8 we observe that each cluster has a different number of programs; hence, the weight assigned to each representative program should depend on the number of programs that it represents.

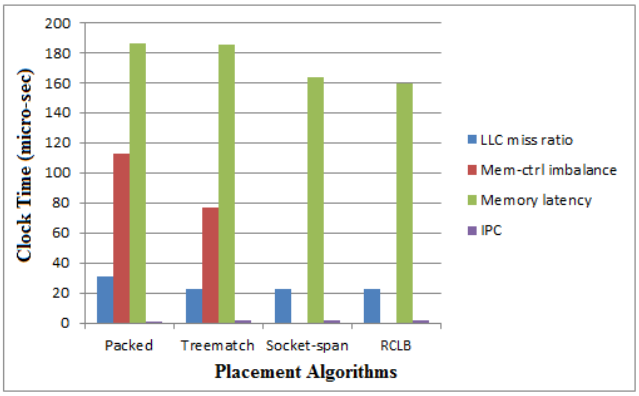

FIGURE 13. Evaluation of simulation entities used to check Memory congestion effect on different CPUs and SPEC benchmarks.

D. STATISTICAL TEST AND COMPARISON

In this section, we have presented a statistical test to analyze the significant difference between proposed and compared methods [56], based on simulation entities and executed on different clusters [1], [7]. First, we have applied the Friedman test [1], [7] to analyze the significant differences in average influence. Since the result shows a significant difference of 1.6 to 13.44 for Treematch(sim), which is known as a benchmark for cluster computing. For Socket-span(sim), we have found a difference in speed up from 0.02 to 2.19, for cluster B we have achieved a difference of 0.04 to 1.8 (shown in table 10). Therefore we have applied the Holm-Bonferroni procedure [12] as post-hoc procedure to find the degree of rejection for each hypothesis (shown in Table-11). We consider the level of confidence $\alpha_c = 0.05$ and the degree of freedom $D_f = 3$ for both.

The Friedman test indicates a significant difference in average influence spread for each iteration, as shown in table-10. The *null* hypothesis (H_0) states that the methods compared are statistically equivalent with no significant difference. The Friedman test rejects the hypothesis because the test statistic value F_f is greater than the $X^2(\alpha_3, D_f)$, i.e. $F_f > 11.0705$. Table 10 shows the results of the Friedman test for average influence spread and indicates that the null hypothesis is rejected for each iteration. Similarly, the post hoc procedure test also rejects the null hypothesis for each iteration. Therefore, the Holm-Bonferroni procedure (independent test statistics) has been applied to measure the concrete differences between all proposed methods. The Holm-Bonferroni procedure rejects the hypothesis because the p-value is greater than the adjusted level of confidence value shown in Table-11. Therefore, we can say that the statistical tests on proposed clusters demonstrate that our

TABLE 10. The Friedman test on Mining erasable item-set MEI and MEIs based on multicore processors pMEI based method for running time.

Cluster Name	Iteration	Execution time-value based on cluster ((Treematch(sim)), (Socket-span(sim))				Test value	State Result
		$Treematch$	$Socket-span$	$FBCA_Cl_A$	$ARRE\_Cl_B$	$F_f\ (d_f = 3)$	Is $F_f > x^2$?
Cluster A	1	5.10997	0.28429	0.43802	0.11229	20.85	Null Hypothesis Rejected
	2	10.44236	0.56932	0.92024	0.22400		
	3	13.44840	1.11182	1.87415	0.44454		
	4	9.59246	1.11644	1.87891	0.44937		
	5	7.86855	0.03612	0.04748	0.01455		
	6	1.65049	0.07156	0.09660	0.02905		
	7	1.61954	0.14217	0.19861	0.05719		
	8	4.89208	2.19081	0.63744	2.17903		
Cluster B	1	4.85260	0.62234	0.62234	0.62234	19.471	Null Hypothesis Rejected
	2	4.94620	0.60050	0.60050	0.60050		
	3	4.85474	0.59273	0.59273	0.59273		
	4	4.86798	0.59418	0.59418	0.59418		
	5	4.10985	0.60091	0.60091	0.60091		
	6	4.79927	0.61573	0.61573	0.61573		
	7	4.90513	0.62120	0.62120	0.62120		
	8	4.91244	0.60606	77.57683	0.60606		

TABLE 11. The estimation of p-value based on Holm procedure (unadjusted) for post-hoc analysis; TM:Treematch, SS:Socket-span.

cluster	p-values								
	Control method=$FFTCl_A$			Control method=$FFTCl_B$			Control method=$FFTCl_C$		
	$TM(sim)$	$TM(real)_B$	$FBCA\_cl$	$SS(sim)$	$SS(real)$	$FBCA\_cl$	$ARRE$	$ProposedCl_A$	$ProposedCl_B$
A	9.20E-07	0.01927	0.00114	0.00015	0.01927	6.30E-06	7.40E-10	0.00114	6.30E-06
B	7.00E-08	0.48	1	8.10E-07	0.48	0.48	7.00E-08	1	0.48

method is comparable to both base algorithms [56] and significantly superior to the remaining existing methods in terms of running time.

VIII. LIMITATIONS AND FUTURE WORK

Based on conducted research work this paper has some limitation and need to work in near future. Some of them are as follows:

- **Assumptions:** The models and algorithms are rely on certain assumptions about the behavior of jobs and resources. If these assumptions do not hold in real-world scenarios, it could impact the accuracy and reliability of predictions.
- **Dependency Modeling and Real-world Implementation:** The Dependency Modeling and prediction is a complex task. So, there may always a chance to explore more about the challenges and limitations of implementing the proposed algorithms in real-world HPC systems.
- **Benchmarking:** The validation of proposed algorithm is based on some workflow scheduling algorithm, there are several workflow scheduling algorithms remains untouched and unexplored.

The further work can be done based on the following parameters to improve and justify more on this research.

- **Auto-tuning of Parameters:** Develop automated techniques for tuning the hyperparameters of the proposed algorithms. This could involve leveraging machine learning or optimization approaches to find the optimal settings for different HPC scenarios, leading to improved overall performance.
- **Scalability Studies:** This research has conducted on Param shivay Super computer having 1.20 Lacs processors and 833 TF speed. Still all the computing nodes has not been used in this research, so there is still a chance available to conduct in-depth scalability studies to assess the performance of the algorithms and models as the size of the HPC system and the complexity of workflows increase.
- **Examine Resource Prediction Framework:** Further explore and refine the resource prediction framework. Consider additional factors and parameters that could impact resource requirements, such as variations in network conditions, and devise strategies to enhance the accuracy of resource predictions.

IX. CONCLUSION

In this paper, we have proposed two algorithms Feature-based capability prediction Algorithm FBCA and Accuracy and Relative Runtime Error ARRE for long-range resource prediction problems. These algorithms are a part of the intelligent ensemble learning technique. The FBCS algorithm is proposed to reduce the noise presented in the datasets whereas the ARRE algorithm is used to improve the model ability so that the model becomes different from the traditional model. The experimental setup utilized eight nodes, each selected from a set of CPU components, incorporating Nehalem CPU cores, Tesla GPU cores, SC10-EP, and SC10-EX configurations. The proposed algorithm demonstrated superior performance in extensive experiments, showcasing its effectiveness in resource prediction and load distribution optimization. Parametric experiments revealed

that the proposed algorithm, under various parameters, achieved optimal performance with a 10-fold cross-validation at k=10. Performance comparison with different feature dimensions showcased stability in accuracy, F-1 score, and absolute root mean error within specific feature dimension ranges. Comparative analysis against traditional models (SVM, DT, KNN, Bagging Ensemble) highlighted the superiority of the proposed algorithm. For instance, the proposed algorithm outperformed SVM and DT with an accuracy of 94.11%, a lower RMSE of 0.06, and a higher F1-score of 94.68%. Comparative evaluation against major resource prediction methods (REAP, BN, LR, RF, FNN) underscored the proposed algorithm's superiority. In comparison to RF, the proposed algorithm achieved an accuracy of 91.82%, a lower RMSE of 0.38, and a higher F1-score of 87.13%. A new SoS and Phold based resource prediction framework has been proposed to describe the simulation model and computes the required resources. The proposed framework has three levels that automatically predict the computing resources that will be used for a long-time. Our Experimental result shows that the proposed algorithm can effectively predict the long-term resource utilization for scientific computing and achieves better accuracy from 8% to 18%. The Validation of created simulation entities, utilizing inter-process communication and L1-cache memory-based miss ratio techniques, aimed to identify suitable simulation entities for modern CPUs and SPEC benchmarks. The Statistical tests, including the Friedman test and post-hoc procedures, confirmed that the proposed algorithm is comparable to base algorithms and significantly superior to other existing methods in terms of running time. For instance, the AREP method applied to ensemble sizes demonstrated a gradual increase in ARE until an ensemble size of 8, indicating saturation beyond that point. We will use one more method based on incremental learning in the near future so that our proposed algorithm will update its predicted model to dynamic changes required during execution.

REFERENCES

[1] N. M. Upadhyay, R. S. Singh, and S. P. Dwivedi, "Prediction of multicore CPU performance through parallel data mining on public datasets," *Displays*, vol. 71, Jan. 2022, Art. no. 102112.

[2] S. Wang, F. Zhu, Y. Yao, W. Tang, Y. Xiao, and S. Xiong, "A computing resources prediction approach based on ensemble learning for complex system simulation in cloud environment," *Simul. Model. Pract. Theory*, vol. 107, Feb. 2021, Art. no. 102202.

[3] V. Mauch, M. Kunze, and M. Hillenbrand, "High performance cloud computing," *Future Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1408–1416, 2013.

[4] C. Xiaomeng, Z. H. Bai, H. Yang, Z. Xujian, and Li Bo, "Runtime prediction of high-performance computing jobs based on ensemble learning," in *Proc. 4th Int. Conf. High Perform. Compilation, Comput. Commun.*, 2020, pp. 56–62.

[5] A. E. Tyas, A. Dharma Wibawa, and M. H. Purnomo, "Theta, alpha and beta activity in the occipital based on EEG signals for mental fatigue in high school students," in *Proc. Int. Conf. Smart Technol. Appl. (ICoSTA)*, Feb. 2020, pp. 1–7.

[6] D. Segev, "An approximate dynamic-programming approach to the joint replenishment problem," *Math. Oper. Res.*, vol. 39, no. 2, pp. 432–444, May 2014.

[7] N. M. Upadhyay and R. S. Singh, "An effective scheme for memory congestion reduction in multi-core environment," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 6, pp. 3864–3877, Jun. 2022.

[8] H. Khaleghzadeh, R. R. Manumachu, and A. Lastovetsky, "A novel data-partitioning algorithm for performance optimization of data-parallel applications on heterogeneous HPC platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 10, pp. 2176–2190, Oct. 2018.

[9] R. Neill, A. Drebes, and A. Pop, "Automated analysis of task-parallel execution behavior via artificial neural networks," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2018, pp. 647–656.

[10] A. Phansalkar, A. Joshi, L. Eeckhout, and L. K. John, "Measuring program similarity: Experiments with SPEC CPU benchmark suites," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Mar. 2005, pp. 10–20.

[11] H. Wei, Z. Wang, G. Hua, J. Sun, and Y. Zhao, "Automatic group-based structured pruning for deep convolutional networks," *IEEE Access*, vol. 10, pp. 128824–128834, 2022.

[12] S. S. Singh, A. Kumar, K. Singh, and B. Biswas, "LAPSO-IM: A learning-based influence maximization approach for social networks," *Appl. Soft Comput.*, vol. 82, Sep. 2019, Art. no. 105554.

[13] P. Arabas, "Modeling and simulation of hierarchical task allocation system for energy-aware HPC clouds," *Simul. Model. Pract. Theory*, vol. 107, Feb. 2021, Art. no. 102221.

[14] S. Li, Y. Li, W. Han, X. Du, M. Guizani, and Z. Tian, "Malicious mining code detection based on ensemble learning in cloud computing environment," *Simul. Model. Pract. Theory*, vol. 113, Dec. 2021, Art. no. 102391.

[15] F. Yao, Y. Yao, L. Xing, H. Chen, Z. Lin, and T. Li, "An intelligent scheduling algorithm for complex manufacturing system simulation with frequent synchronizations in a cloud environment," *Memetic Comput.*, vol. 11, no. 4, pp. 357–370, Dec. 2019.

[16] G. Kaur, A. Bala, and I. Chana, "An intelligent regressive ensemble approach for predicting resource usage in cloud computing," *J. Parallel Distrib. Comput.*, vol. 123, pp. 1–12, Jan. 2019.

[17] R. Shaw, E. Howley, and E. Barrett, "An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions," *Simul. Model. Pract. Theory*, vol. 93, pp. 322–342, May 2019.

[18] X.-T. Nguyen, T.-T. Hoang, H.-T. Nguyen, K. Inoue, and C.-K. Pham, "An FPGA-based hardware accelerator for energy-efficient bitmap index creation," *IEEE Access*, vol. 6, pp. 16046–16059, 2018.

[19] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *J. Netw. Comput. Appl.*, vol. 82, pp. 93–113, Mar. 2017.

[20] N. An, H. Ding, and J. Yang, "Deep ensemble learning for Alzheimer's disease classification," *J. Biomed. Informat.*, vol. 105, Oct. 2020, Art. no. 103411.

[21] H. Wang, Y.-M. Zhang, J.-X. Mao, and H.-P. Wan, "A probabilistic approach for short-term prediction of wind gust speed using ensemble learning," *J. Wind Eng. Ind. Aerodyn.*, vol. 202, Jul. 2020, Art. no. 104198.

[22] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, "Hashing with mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2424–2437, Oct. 2019.

[23] T. A. Engel, A. S. Charão, M. Kirsch-Pinheiro, and L.-A. Steffenel, "Performance improvement of data mining in Weka through GPU acceleration," *Proc. Comput. Sci.*, vol. 32, pp. 93–100, Jan. 2014.

[24] M. S. E. Ipek, B. R. De Supinski, and S. A. McKee, "An approach to performance prediction for parallel applications," in *Proc. Eur. Conf. Parallel Process.* Cham, Switzerland: Springer, 2005, 196–205.

[25] G. K. Shyam and S. S. Manvi, "Virtual resource prediction in cloud environment: A Bayesian approach," *J. Netw. Comput. Appl.*, vol. 65, pp. 144–154, Apr. 2016.

[26] B. C. Lee and D. M. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," in *Proc. 12th Int. Conf. Architectural Support Program. Lang. Operating Syst. (ACM)*, 2006, pp. 185–194.

[27] S. Sharkawi, D. DeSota, R. Panda, R. Indukuru, S. Stevens, V. Taylor, and X. Wu, "Performance projection of HPC applications using SPEC CFP2006 benchmarks," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, May 2009, pp. 1–12.

[28] R. M. Fujimoto, "Research challenges in parallel and distributed simulation," *ACM Trans. Model. Comput. Simul.*, vol. 26, no. 4, pp. 1–29, May 2016.

[29] X. E. Chen and T. M. Aamodt, "Hybrid analytical modeling of pending cache hits, data prefetching, and MSHRs," *ACM Trans. Archit. Code Optim.*, vol. 8, no. 3, pp. 1–28, Oct. 2011.

[30] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith, "A mechanistic performance model for superscalar out-of-order processors," *ACM Trans. Comput. Syst.*, vol. 27, no. 2, pp. 1–37, May 2009.

[31] S. Eyerman, K. Hoste, and L. Eeckhout, "Mechanistic-empirical processor performance modeling for constructing CPI stacks on real hardware," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Apr. 2011, pp. 216–226.

[32] A. Hartstein and T. R. Puzak, "The optimum pipeline depth for a microprocessor," in *Proc. 29th Annu. Int. Symp. Comput. Archit.*, 2002, pp. 7–13.

[33] S. Van den Steen, S. De Pestel, M. Mechri, S. Eyerman, T. Carlson, D. Black-Schaffer, E. Hagersten, and L. Eeckhout, "Micro-architecture independent analytical processor performance and power modeling," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2015, pp. 32–41.

[34] S. Van den Steen, S. Eyerman, S. De Pestel, M. Mechri, T. E. Carlson, D. Black-Schaffer, E. Hagersten, and L. Eeckhout, "Analytical processor performance and power modeling using micro-architecture independent characteristics," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3537–3551, Dec. 2016.

[35] G. M. Weber, "MATOG: Array layout auto-tuning for CUDA," *ACM Trans. Archit. Code Optim.*, vol. 14, no. 3, p. 28, 2017.

[36] K. Singh, E. Ipek, S. A. McKee, B. R. de Supinski, M. Schulz, and R. Caruana, "Predicting parallel application performance via machine learning approaches," *Concurrency Comput., Pract. Exper.*, vol. 19, no. 17, pp. 2219–2235, Dec. 2007.

[37] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang, "CloudProphet: Towards application performance prediction in cloud," in *Proc. ACM SIGCOMM Conf.*, Aug. 2011, pp. 426–427.

[38] N. Ardalani, C. Lestourgeon, K. Sankaralingam, and X. Zhu, "Cross-architecture performance prediction (XAPP) using CPU code to predict GPU performance," in *Proc. 48th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2015, pp. 725–737.

[39] X. Zheng, L. K. John, and A. Gerstlauer, "Accurate phase-level cross-platform power and performance estimation," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2016, pp. 1–6.

[40] X. Zheng, H. Vikalo, S. Song, L. K. John, and A. Gerstlauer, "Sampling-based binary-level cross-platform performance estimation," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 1709–1714.

[41] B. Piccart, A. Georges, H. Blockeel, and L. Eeckhout, "Ranking commercial machines through data transposition," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Nov. 2011, pp. 3–14.

[42] A. Phansalkar, A. Joshi, and L. K. John, "Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite," in *Proc. 34th Annu. Int. Symp. Comput. Archit.*, Jun. 2007, pp. 412–423.

[43] A. Phansalkar, A. Joshi, and L. K. John, "Subsetting the SPEC CPU2006 benchmark suite," *ACM SIGARCH Comput. Archit. News*, vol. 35, no. 1, pp. 69–76, Mar. 2007.

[44] D. Shelepov, J. C. Saez Alcaide, S. Jeffery, A. Fedorova, N. Perez, Z. F. Huang, S. Blagodurov, and V. Kumar, "HASS: A scheduler for heterogeneous multicore systems," *ACM SIGOPS Operating Syst. Rev.*, vol. 43, no. 2, pp. 66–75, Apr. 2009.

[45] D. Shepelow and A. Fedorova, "Scheduling on heterogeneous multicore processors using architectural signatures," in *Proc. WIOSCA Workshop 35th Annu. Int. Symp. Comput. Archit.*, 2008, pp. 1–9.

[46] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-aware scheduling for heterogeneous datacenters," in *Proc. 18th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, vol. 48, 2013, pp. 77–88.

[47] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and QoS-aware cluster management," in *Proc. 19th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2014, pp. 127–144.

[48] K. Hoste, L. Eeckhout, and H. Blockeel, "Analyzing commercial processor performance numbers for predicting performance of applications of interest," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, Jun. 2007, pp. 375–376.

[49] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L. K. John, and K. De Bosschere, "Performance prediction based on inherent program similarity," in *Proc. Int. Conf. Parallel Architectures Compilation Techn. (PACT)*, Sep. 2006, pp. 114–122.

[50] S. Haas, T. Karnagel, O. Arnold, E. Laux, B. Schlegel, G. Fettweis, and W. Lehner, "HW/SW-database-codesign for compressed bitmap index processing," in *Proc. IEEE 27th Int. Conf. Application-specific Syst., Architectures Processors (ASAP)*, Jul. 2016, pp. 50–57.

[51] N.-P. Tran, M. Lee, and D. H. Choi, "Memory-efficient parallelization of 3D lattice Boltzmann flow solver on a GPU," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. (HiPC)*, Dec. 2015, pp. 315–324.

[52] M. Nelson, Z. Sorenson, J. M. Myre, J. Sawin, and D. Chiu, "GPU acceleration of range queries over large data sets," in *Proc. 6th IEEE/ACM Int. Conf. Big Data Comput., Appl. Technol.*, Dec. 2019, pp. 11–20.

[53] A. Lastovetsky and R. R. Manumachu, "New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1119–1133, Apr. 2017.

[54] R. Huang, L. Ma, J. He, and X. Chu, "T-GAN: A deep learning framework for prediction of temporal complex networks with adaptive graph convolution and attention mechanism," *Displays*, vol. 68, Jul. 2021, Art. no. 102023.

[55] B. Huynh and B. Vo, "An efficient method for mining erasable itemsets using multicore processor platform," *Complexity*, vol. 2018, pp. 1–9, Oct. 2018.

[56] D. Mejri, M. Limam, and C. Weihs, "A new dynamic weighted majority control chart for data streams," *Soft Comput.*, vol. 22, no. 2, pp. 511–522, Jan. 2018.

[57] J. Li, G. Michelogiannakis, B. Cook, D. Cooray, and Y. Chen, "Analyzing resource utilization in an HPC system: A case study of NERSC's perlmutter," in *High Performance Computing* (Lecture Notes in Computer Science), vol. 13948. Cham, Switzerland: Springer, 2023.

NAVIN MANI UPADHYAY received the B.E. and M.Tech in CSE. He is a Ph.D. scholar in computer science and engineering at IIT(BHU) Varanasi, India. His research focuses on high-performance computing, particularly in advancing computational capabilities.

RAVI SHANKAR SINGH (Senior Member, IEEE) received the B.Tech., M.Tech., and Ph.D. degrees in CSE.

He is currently an Associate Professor with the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, Varanasi. More than 50 research papers has included in his account. His research interests include high performance computing and cloud computing.

Dr. Singh is a Senior Member of ACM.

SHRI PRAKASH DWIVEDI received the B.Tech. degree in IT, the M.Tech. degree in CS and the Ph.D. degree in CSE. He is currently an Assistant Professor with the Department of IT, G. B. Pant University of Agriculture Technologies, Pantnagar, India. His current research interests include algorithms, graph matching, and pattern recognition.