VEERBench – an intelligent computing framework for workload characterisation in multi-core heterogeneous architectures

Yokesh Babu Sundaresan* and M.A. Saleem Durai

School of Computer Science and Engineering Professors – SCOPE, VIT University, Vellore, India
Email: yokeshbabu.s@vit.ac.in
Email: masaleemdurai@vit.ac.in
*Corresponding author

Abstract: Multi-core heterogeneous architectures are playing the important role in server, mobile and all commercial devices. With the advent of internet of things in today’s applications and increase in the workloads inputs, predictive simulating and computing tools are mandatory for the effective implementation of the multi-core heterogeneous architectures for the different applications. Many tools such as MacPACT, ESEC has been into existence but an intelligent computing framework tool for the predictive selection of the cores depending on the workloads remains in the darker side of the research. Hence the new computing framework called VEERBENCH has been proposed which works on the learning and training mechanisms for the usage of the cores in the heterogeneous architectures depending on the workloads. The framework uses the fuzzy clustering with the extreme learning machines and formulation of adaptive and cognitive energy (FACE) rule sets which are used for the energy and performance-based allocation of the cores. The proposed knowledge-based test bench has been compared with the other tools such as MACPACT, ESEC and with the other energy-based scheduler benchmarks and the obtained results are shown.

Keywords: VEERBENCH; extreme learning machines; ELM; formulation of adaptive and cognitive energy; FACE; MacPACT; ESEC; internet of things.


Biographical notes: Yokesh Babu Sundaresan is currently working as an Assistant Professor (Selection Grade) in School of Computer Science and Engineering (SCOPE) at VIT, Vellore. He has around six years of industrial experience on hardware. His research interests include heterogeneity in embedded systems, low power embedded systems, wireless sensor networks, reconfigurable architectures and cloud computing having published ten papers in peer-reviewed journals.
VEERBENCH

M.A. Saleem Durai is currently working as an Associate Professor in the School of Computer Science and Engineering (SCOPE) at VIT, Vellore. He has around ten years of experience in academics. His research interests include cloud computing and computer networks having more than 20 papers in peer-reviewed journals.

1 Introduction

Multi-core heterogeneous architectures are used in many applications starting from the smart home automation to the high end mobiles and robotics applications. Since it finds the most of the applications, the knowledge of using the multi-core heterogeneous architectures is still under darker side because of its difficulty in the implementation, programming structures, increased number of the workloads and its complexity in the architectures.

The workloads are increased day by day in proportional to the complexity and the applications of the multi-core heterogeneous architectures. In the recent world of the Internet of Things, the workloads increases day by day which makes the users in the complex conditions for the effective implementation. Hence the workloads are to be characterised accordance to the architectures for the less energy consumptions and high performance mechanisms.

For the above constraints in multi-core architectures, new computing framework called versatile energy elegant reliable (VEERBENCH) has been proposed. The proposed workbench works on the principle of extreme learning machines (ELM) with the new set of the rule sets called formulation of the adaptive cognitive energy (FACE) has been used in the framework for the intelligent allocation and prediction of the cores according to the workloads characterisation. The behind screen algorithms are being compared with the fuzzy intelligent algorithms suggested by Cerotti et al. (2016) and the proposed algorithm proves to be more vital for the computing the workloads.

The proposed computing framework has been tested with the SPEC-2000 bench marks, SPLASH, along with the different multi-core heterogeneous architectures. The computing framework will be new visualisation tools which imparts the intelligent classification of the cores for the effective functioning of the architectures and also in sights the predictive view of the cores in accordance with the workloads.

This paper initially discusses related works with the proceeding sections on the proposed algorithms, framework implementation and finally with the results and performance evaluation.

2 Related works

Chen and John (2008) proposed an energy-aware scheduling mechanism that employs fuzzy logic to calculate the suitability between programs and cores by analysing important inherent program characteristics such as instruction dependency distance and branch transition rate. The obtained suitability is then used to guide the program scheduling in the heterogeneous multi-core system. Accomplished a few outcomes that
shows suitability-guided program scheduling mechanism achieves up to 15.0% average reduction in energy-delay product compared with that of the random planning approach.

Chen and John (2009) introduced a technique to leverage the inherent characteristics of a program for scheduling decisions in heterogeneous multi-core processors. The proposed strategy projects the core’s configuration and the program’s resource demand to a unified multi-dimensional space and uses weighted Euclidean distance between these two to guide the program scheduling. The outcomes of the proposed system show that on average, this distance-based scheduling heuristic achieves 24.5% reduction in energy delay product, 6.1% decrease in energy and 9.1% improvement in throughput when compared with traditional hardware unaware planning calculation.

Cerotti et al. (2016) proposed two approaches to characterise multi-threaded applications in multi-core environments described by a limited number of parameters. They used wider modelling technique that aims to characterise generalised performance metrics of multi-core CPU with a limited complexity. They worked on how such parameters can be derived from measurements originating from executions of real applications on real multi-core machines have been given.

Shelepov et al. (2010) introduced a heterogeneity-aware signature-supported scheduling algorithm that does the coordinating using per-thread architectural signatures, which are compact summaries of threads’ architectural properties collected offline. The resulting algorithm does not depend on dynamic profiling and is similarly basic and adaptable. We actualised HASS in open Solaris and accomplished average workload speedups of up to 13%, matching best static task, achievable only by an oracle. Thus HASS is able to achieve performance comparable to the best (oracle) static assignment. Contrary to our expectations, we found that lack of phase awareness in HASS has attempted to its advantage, because it saved it from many problems linked to phase changes that were uncovered by our implementation of IPC-driven algorithm.

3 Proposed framework VEERBENCH – overview and working mechanism

The test bench simulator’s has the three different frameworks for the predictive analysis of the cores in accordance with the energy and performance. The simulator engine has the following phases of working:

Phase 1 Calculation and detection phases.

Phase 2 Prediction phase.

In Phase 1 of the calculation, branch suitability, cache memory metrics, energy, performance are calculated and detected. In Phase 2, intelligent prediction of the energy in accordance to the usage of the workloads and the architectures will be detected. The framework engine works on the combination of the clustering, computing and decision which is shown in Figure 1.
Figure 1 VEERBENCH – framework and working mechanism of the intelligent computing of the energy in accordance with the cores

3.1 Stage 1: clustering of the workloads

Stage 1 involves around the clustering of the workloads in which the workloads are clustered depending on their characteristics such as fixed point clusters and the floating point clusters as shown in Figure 2. All the sensor inputs with digitisation are considered as the Fixed point and whereas all the video, image inputs are scaled to the floating point clusters. The two separate clusters are formed in which the energy, CPU performance, speed and the branch fitness function (BFF), ILP fitness functions will be calculated according to the workloads and for accurate computing the resultant values are feed into the learning machines.

Figure 2 Clustering of the workloads based on their characteristics which yield the two clustering types (see online version for colours)
The learning machines used in the framework are based on the extreme learning machines which are most widely used for its speed of learning and its accuracy. These extreme learning machines are single feed forward networks used for the classification or regression. It consist of the randomly assigned and less updating of the single layer of hidden nodes to which the inputs are connected. The weights of the hidden layers and the outputs are learned in the single step which leads to the fastest simulation for the better accuracy.

After the ELM Phases, FACE rules are used for the prediction and allocation of the core allocation depends on the energy calculated and also with the other parameters like performance, BFF, ILP etc. Also with the help of the above tools, runtime calculation of the energy in the core will be determined.

4 Parameters calculation

The VEERBENCH framework calculates the different phases of the calculation such as energy phases, branch fitness function, performance and the ILP measurement mechanisms. The measurement phases are explained as follows.

4.1 Energy factors mechanism

The energy is calculated into two phases. In the first stage, energy is calculated after the clustering of the workloads. This energy calculated workloads are act as the inputs to the extreme learning machines. The clustering of the ‘n’ workloads depends on measuring its MIPS in which leads to the big loads, medium loads, small loads (fixed point). The fuzzy-based principle of clustering principle is used for the classification.

Let us consider the \( n_1, n_2, n_3, n_4 \ldots n_n \) be the number of the workloads with the clock frequency \( F_c \), in which the energy is calculated as follows:

\[
\text{Total energy to be calculated for the workloads: } E_n = P \sum n \times F_c
\]  

(1)

Based on the total energy calculated different energy membership functions are calculated for the clustering which is given as follows:
Step 1  total energy is subdivided into the scales of the energies depending on the workloads which are as follows:

\[ \text{Average scale of threshold east} = \frac{E_n}{n} \]  

(2)

Step 2  based on the AST, different groupings of the energy have been calculated depends on the

\[ E(BL) = E1 \text{ where East} > E1 < En \]  

(3)

\[ E(ML) = E2 \text{ where East} = E2 < En \]  

(4)

\[ E(SL) = E3 \text{ where } E3 < \text{ East} \]  

(5)

After the calculation of the energies of each load, these are fed to the next phases of the Intelligent computing phases which uses the ELM as the main engine for which the Energies are obtained as the accurate determination.

Once the energy of the workloads is calculated accurately, the framework uses the FACE rule sets in which the cores are allocated depend on the energy of the workloads.

4.2 Branch fitness function

The branch fitness function is the features of the proposed framework which tries to measure the match between workload’s branch predictability and the branch predictor size. To manipulate the branch fitness of the program, branch transition rate are used, which is demonstrated to be an appropriate metric for the branch predictability of the program by Huang et al. (Kumar et al., 2004) which is used by the proposed framework.

The framework uses the principle of transition rates determination with the very high and low transition rates are used as the threshold for the branch suitability and predictability.

Let us consider the ten clusters with the transition values ranges from the 0.0 to 1. For each workload, branch instruction falls between any one of the 10 clusters with the transition rates falls between the 0 to 0.1 and 0.8 to 1 can be best predictors of the branch transition rates. As the transition rates are reaches 50%, it is hard to predict and hence with the very low and very high transition rates are considered. With the branch suitability can be determined by the expression (6).

\[
B(F) - Bi = \left[ \sum_{n=0}^{N} Bn \ast \left( \sum_{n=0}^{N} Pn + P_{n+1} \right) \ast A \ast \sum_{n=1}^{N+n+1} Pn \right]
\]

(6)

where:

- \( B(F) \)  branch fitness function
- \( Bi \)  size of the branch predictors ranging from 1 … N
- \( A \)  tuning factor
- \( P_i \)  conditional branch equations present in the workloads.
4.3 ILPcache size estimators

The cache size estimators estimate the cache local size of the architectures with the program size. It is being estimated with the no of access of the program in the memory by introducing the parameters called cache distance estimation metrics which is given by the expression (7)

\[
\text{Cache size estimation} = \frac{PD}{C}
\]

where \(C\) is the L1 data cache size and \(PD\) is the percentage of the data accesses with reuse distance less than \(C\). The values of accessing the memory is either 0 or 1, the overall cache size estimation is given by the expression (8)

\[
C(E) = \left( \frac{P_c < C_m}{P_c < C_m} \right)_\text{max}
\]

where:

- \(C(E)\) cache estimation process
- \(C_m\) no. of cache in the multi-core heterogeneous architectures.

4.4 Performance predictors

The performance predictors are the final estimators of the framework which calculates the performance of the architecture and gives the insight view of the multi-core architectures based on the above mentioned parameters. This phase decides the calculation and prediction of the cores in accordance with the different parameters estimated based on the intelligent extreme learning machines and FACE rule sets. The mathematical expression of the performance predictors (PP) value is given by the expression (9).

5 Intelligent extreme learning machines with the face rule sets

The intelligent extreme learning machines are based on the principle of the extreme learning machines along with the FACE rule sets. These machines along with the adaptive cognitive rule sets predict the usage of the cores according to the parameters measured.

5.1 Extreme learning machines

These machines are characterised with the single brain like neural networks in which the input weights and hidden layers are randomly chosen and also need not to be tuned mandatorily. This is considered to be linear system which analytically determines the output weights with the high speed and high accuracy.
Figure 4  Basic principles of extreme learning machines with the transition weights, inputs and output weights

The general mathematical model for the general mathematical function is given as follows:

$$H\beta = T.$$  

where

$$H(w_1, \ldots, w_N, b_1, \ldots, b_N, x_1, \ldots, x_N) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_N \cdot x_1 + b_N) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_N \cdot x_N + b_N) \end{bmatrix}_{N \times N}.$$  

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_N^T \end{bmatrix}_{N \times m} \quad \text{and} \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}.$$  

5.2 ELM algorithm for the proposed framework

The algorithm has been implemented for the different parameters which are measured based on the workloads characterisation given as follows:

1. consider the energy training sets, branch training sets and cache estimators has the input matrices with the activation function $h(x)$ and with the hidden layers of $B$

2. randomly assign the input weights $J$ and the bias $F$

3. calculate the hidden layers and the output matrices and also calculate the output weight based on the inputs which are given as:

   a. Energy learning mechanism:
      $$F = B'(E)T$$
      where $T$ – transition matrices.
For branch and cache learning mechanism:

\[ F = B'(B/C)T \]

where \( T \) – transition matrices.

### 5.3 Formulation of the cognitive additive rule sets

After learning and getting the optimised parameters in depends on the workloads using the extreme learning machines, we formulate the prediction factor called \( p_f \) in which are used for the allocation of the cores in accordance with the different parameters. The FACE rule sets can be formulated as follows.

Let us consider the four cores in which the energy of the cores is calculated as the \( E_{c1}, \ E_{c2}, \ E_{c3} \) and \( E_{c4} \) respectively. The energy of the applications/workloads are classified as the \( e_{t1}, \ e_{t2}, \ e_{t3} \) and \( e_{t4} \). By assuming the conditions \( E_{c1} < E_{c2} > E_{c3} < E_{c4} \) for the first heterogeneous core, allocation of the cores/prediction of the cores in depends on the energy calculated, branch fitness function and cache size estimation is given in Table 1.

**Table 1** FACE rule sets for the allocation of the cores/prediction depending on the workload parameters

<table>
<thead>
<tr>
<th>Sl. no.</th>
<th>Metrics taken</th>
<th>Core’s characteristics</th>
<th>FACE rule sets</th>
<th>Allocation of the cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Energy of the workloads (( E_{t1} \ldots E_{tn} ))</td>
<td>( E_{c1} ) ( E_{c2} ) ( E_{c3} ) ( E_{c4} )</td>
<td>IF (( E_{t1} &lt; E_{c1} )) and (( E_{t1} &lt; E_{c2} )) and (( E_{t3} &lt; E_{c3} )) and (( E_{t4} &lt; E_{c4} ))</td>
<td>Core C1 is allocated for the workloads and other cores as OFF</td>
</tr>
<tr>
<td>2</td>
<td>Energy of the workloads, branch metrics</td>
<td>( C_1 ) with the 1K share (( E_{b1} )) ( C_2 ) with 2K share (( E_{b2} )) ( C_3 ) with 3K share (( E_{b3} )) ( C_4 ) with 4K share (( E_{b4} ))</td>
<td>IF (( E_{t1} &lt; E_{b1} )) and (( E_{t1} &lt; E_{b2} )) and (( E_{t3} &lt; E_{b3} )) and (( E_{t4} &lt; E_{b4} ))</td>
<td>Core C1 is allocated for the workloads and other cores as OFF</td>
</tr>
<tr>
<td>3</td>
<td>Energy of the workloads, branch metrics, cache size</td>
<td>( C_1 ) with the 1K share with 8 KB cache (( E_{a1} )) ( C_2 ) with 2K with 16 KB cache (( E_{a2} )) ( C_3 ) with 3K share with 32 KB cache (( E_{a3} )) ( C_4 ) with 4K share (( E_{a4} )) with 64 KB cache (( E_{a4} ))</td>
<td>IF (( E_{t1} &lt; E_{a1} )) and (( E_{t1} &gt; E_{a2} )) and (( E_{t3} &lt; E_{a3} )) and (( E_{t4} &lt; E_{a4} ))</td>
<td>Core C1 is allocated for the workloads and other cores as OFF</td>
</tr>
</tbody>
</table>
6 Overall flow of the framework

Figure 5 Shows the overall flow of the framework of the proposed VEERBENCH mechanism
7 Experimental setup

For the experimenting the framework, we have used the different multi-core architectures with the following specifications which are as follows as

<table>
<thead>
<tr>
<th>Sl. no.</th>
<th>Core's type tested Core C1</th>
<th>Core C2</th>
<th>Core C3</th>
<th>Core C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cortex-A5 series ARM-9 series Cortex-A5 series Cortex-A8 series</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ARM-9 TDMI series ARM-9 TDMI series Cortex-R series Cortex A5 series</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Cortex-A5 series CortexA-5 series Programmable FPGA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Cortex A series Programmable FPGA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The framework uses the ARM families of the architectures in the heterogeneous environment with all the characteristics features like energy, cache size, bus interfaces has been taken into an account. The computing platform has been formulated with the MATLAB toolboxes of version of greater than R2012. Also the view of the VEERBENCH has been given in Figure 6.

Figure 6 VEERBENCH framework designed for the multi-core heterogeneous architectures (see online version for colours)

8 Performance evaluation and analysis

The performance of the VEERBENCH framework has been compared with the other exiting tools/schedulers with other exiting algorithms which are given in the preceding sections. The benchmarks such as SPEC benchmark 2010, PARSEC and SPLASH-2. Also the different application workloads such as equake, gcc, mgrid, mesa, art, vpr,
applu, soplex, twolf, wupwise, perlbench, swim, crafty, povray, milc, vortex, libquantum, mcf, gap, leslied, bwaves, lbm astar, sphinx, ackscholes, bodytrack, canneal, facesim, ferret, fluidanimate, swaptions, ocean, ftt, fmm, radix for the testing.

As we have taken the workloads again the performance has been evaluated for the different test multi-core architectures which are explained in Table 2.

8.1 Accuracy

The accuracy of the framework has been measured with the different parameters taken in the workload by using the different intelligent algorithms such as such as fuzzy logic algorithms. The term of accuracy measurement of the framework has been sub-divided into energy, BFF, cache size and the overall performance system along with the four different multi-core architectures.

Figure 7 Energy measurement accuracy for the implementation of fuzzy logics for the multi-core heterogeneous testbed_1 (see online version for colours)

Figure 8 Energy accuracy estimation using the ELM + FACE rule sets in the proposed framework multi-core heterogeneous testbed_1 (see online version for colours)
Figure 9  Energy measurement accuracy for the implementation of fuzzy logics for the multi-core heterogeneous test beds (see online version for colours)

![Figure 9](image)

Figure 10  Energy accuracy estimation using the ELM + FACE rule sets in the proposed framework multi-core heterogeneous test beds (see online version for colours)

![Figure 10](image)

8.2 Branch fitness function evaluation

Again the branch fitness function are evaluated by using the above expression (6) and the comparative results of the implementing the fuzzy and ELM + FACE rule sets are given in Figures 11–14.

Figure 11  BFF accuracy measurement for the fuzzy-based estimation (see online version for colours)

![Figure 11](image)
Figure 12  BFF accuracy estimation of the proposed algorithms with the ELM + FACE rule sets (see online version for colours)

Figure 13  BFF accuracy estimation for the implementation of fuzzy logics for the multi-core heterogeneous test beds (see online version for colours)

Figure 14  BFF accuracy estimation of the proposed algorithms with the ELM + FACE rule sets for the multi-core heterogeneous test beds (see online version for colours)
8.3 Cache size and ILP accuracy estimation

The cache size and ILP accuracy has been estimated by using the expression (8) and tested with the fuzzy logics and proposed framework of ELM + FACE rule sets.

**Figure 15** Cache size estimation using the fuzzy logic algorithm (see online version for colours)

**Figure 16** Cache size estimation using ELM + FACES methods in the proposed algorithms (see online version for colours)

**Figure 17** Cache size estimation using the fuzzy logic algorithm for the multi-core heterogeneous test beds (see online version for colours)
9 Speed calculation and analysis

The proposed framework has been working on the MATLAB environment. The framework’s speed and simulation time are calculated based on the learning and testing time period of the different parameter metrics measurement also with the prediction time and the allocation of the cores in accordance with the workloads.

<table>
<thead>
<tr>
<th>Details of the workloads</th>
<th>Learning phase(s)</th>
<th>Testing phase(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Energy calculation</td>
<td>BFF function</td>
</tr>
<tr>
<td></td>
<td>Cache size estimators</td>
<td>BFF function</td>
</tr>
<tr>
<td></td>
<td>Cache size estimators</td>
<td></td>
</tr>
</tbody>
</table>

### Table 3
Simulation time for the different phases of the framework with the different bench works

<table>
<thead>
<tr>
<th>Details of the workloads</th>
<th>Learning phase(s)</th>
<th>Testing phase(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Energy calculation</td>
<td>BFF function</td>
</tr>
<tr>
<td></td>
<td>Cache size estimators</td>
<td>BFF function</td>
</tr>
<tr>
<td></td>
<td>Cache size estimators</td>
<td></td>
</tr>
</tbody>
</table>

| SPLASH-2   | 10  | 10  | 10  | 08  | 08  | 08  |
| SPEC-2     | 15  | 15  | 15  | 06  | 04.34 | 02.34 |
| PARSEC     | 20  | 12  | 15  | 09  | 4.56  | 4.66  |

10 Overall comparison

VEERBENCH has been compared with the other existing tools/frameworks available and the features are listed and compared as shown in Table 4.

### Table 4
Different characteristics features of the proposed framework and its comparison

<table>
<thead>
<tr>
<th>Characteristics taken</th>
<th>VEERBENCH</th>
<th>MACPACT</th>
<th>ESEC test benches</th>
<th>Other energy-based schedulers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intelligent algorithm imparted</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Energy calculation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Branch predictors</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note: Have been compared with the existing tools and it clearly shows that the more features are implemented in the framework for the better performance calculation
Table 4  Different characteristics features of the proposed framework and its comparison (continued)

<table>
<thead>
<tr>
<th>Characteristics taken</th>
<th>VEERBENCH</th>
<th>MAcPACT</th>
<th>ESEC test benches</th>
<th>Other energy-based schedulers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache and ILP size estimators</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Workloads-core allocation</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Predictive relationship of workload-cores</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Simulation time</td>
<td>Medium</td>
<td>Less</td>
<td>Less</td>
<td>Low–medium</td>
</tr>
<tr>
<td>Relationship between performance-workloads</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Suitability for the different architectures</td>
<td>ARM + FPGA</td>
<td>GPU</td>
<td>GPU</td>
<td>GPU</td>
</tr>
<tr>
<td>Scalability</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Flexibility for run time analysis</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note: Have been compared with the existing tools and it clearly shows that the more features are implemented in the framework for the better performance calculation

11 Conclusions

In this paper, increase in workload inputs on multi-core architecture due to advent in IOT is highlighted and need for predictive simulation and computing tools for effective implementation is emphasised. An intelligent computing framework called VEERBENCH is proposed for workload characterisation and predictive core allocation. VEER is a visualisation computing framework used for the workload-performance relationship on multi-core heterogeneous architectures for the users. It has been tested with the different families of the architectures with ARM as the major constraint for testing. Intelligent framework with the prediction mechanism makes proper computing of the workloads before using in multi-core architectures. The proposed framework is compared with other tools and energy-based schedulers and the results obtained were shown. This framework can be further improvised by adding application architectures and input feed.

References


