International Journal of Science, Strategic Management and Technology
Volume **02** Issue **03** March-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Hardware-Aware Bug Prediction:Leveraging Performance Counters for Early Software Defect Detection

Mrs. Vanitha, Assistant Professor, Department of Computer Technology, Dr.N.G.P. Arts and Science College, Coimbatore, E-Mail: vanitha.p@drngpasc.ac.in

TC. Vivek, Student, Department of Computer Technology, Dr.N.G.P.Arts and Science College,Coimbatore,E-Mail:tcvivek875@gmail.com

https://doi.org/10.55041/ijsmt.v2i3.302

Cite this Article: Vivek, T. (2026). Hardware-Aware Bug Prediction:Leveraging Performance Counters for Early Software Defect Detection. International Journal of Science, Strategic Management and Technology, *02*(03). https://doi.org/10.55041/ijsmt.v2i3.302

Abstract

Software bugs present major challenges in software engineering, resulting in increased costs, security vulnerabilities, and system failures. Traditional bug prediction methods focus on source code metrics and historical defect data, often missing low-level hardware characteristics that reveal performance anomalies indicating latent defects. This paper introduces a Hardware-Aware Bug Prediction System (HABPS) using hardware performance counters (HPCs) for software defect detection. Our approach extracts dynamic hardware-level features during execution, including cache miss rates, branch prediction accuracy, pipeline stalls, and memory access patterns. We evaluate machine learning models—Random Forest, Gradient Boosting, and Neural Networks—trained on HPC data from buggy and clean executions. Experimental results on real-world projects demonstrate 92.3% prediction accuracy with an F1-score of 0.91, outperforming traditional methods by 18.5%. The system identifies defect-prone modules early, reducing debugging effort by approximately 40%.This research connects low-level hardware behaviour with high-level software quality assessment, offering a new paradigm for early bug prediction in complex systems.

Keywords: Bug prediction, machine learning, performance counters, software quality, defect detection.

1 Introduction

Software defect prediction is critical in software engineering. As systems grow in complexity, post-release bug fixes become exponentially more expensive. Traditional bug prediction techniques analyse the source code metrics like cyclomatic complexity or use historical defect data. While successful, they often miss runtime behaviour and low-level execution characteristics indicating latent defects.

Modern processors include hardware performance counters (HPCs) tracking microarchitectural events with minimal overhead. These counters provide execution insights including cache utilization, branch prediction efficiency, memory usage, and instruction-level parallelism. Abnormal patterns in these metrics can signal defects not apparent in static analysis.

This paper presents a Hardware-Aware Bug Prediction System (HABPS) connecting hardware-level execution characteristics with software defect prediction. Key contributions include:

• Framework for collecting and analysing hardware performance counter data • Feature extraction

techniques transforming raw HPC data into defect predictors

• Evaluation of machine learning models using HPC based features

• Comparative analysis showing superiority over code metric-based approaches

• Guidelines for integrating hardware-aware bug prediction into development workflows

2 Related Work

2.1 Traditional Bug Prediction

Early research focused on static code metrics: McCabe's cyclomatic complexity, Halstead's software science metrics, and CK object-oriented metrics. Later approaches added historical defect data, process metrics, and developer information. Machine learning techniques including decision trees, support vector machines, and neural networks applied to these feature sets achieved 70-85% accuracy.

2.2 Hardware-Based Analysis

Hardware performance counters are used for performance optimization and debugging. Previous research showed correlations between HPC readings and specific bug types, particularly performance bugs, memory issues, and concurrency problems. However, these studies focused on specific bug categories rather than comprehensive defect prediction.

2.3 Machine Learning in SE

Machine learning is increasingly applied to software engineering tasks including code completion, test generation, and quality prediction. Recent works explored deep learning for bug prediction using code embeddings and abstract syntax trees. Our work differs by focusing on dynamic, hardware-level features complementing existing approaches.

3 Methodology

3.1 Hardware Counter Selection

We selected 32 hardware performance counters across five categories:

• **Cache Performance:** L1/L2/L3 cache misses, cache references, access patterns

• **Branch Prediction:** Branch mispredictions, conditional branches, indirect branches

• **Memory System:** Memory loads/stores, TLB misses, page faults

• **CPU Pipeline:** Instructions per cycle, pipeline stalls, resource conflicts

• **Energy Consumption:** CPU energy usage, thermal metrics

3.2 Feature Extraction

Let
$H = \{h_1, h_2, \ldots, h_n\}$
represent raw HPC readings. We extract features using:

$$F_i = \frac{1}{T}\sum_{t=1}^{T} h_i(t \quad) \qquad (1)$$

for mean values, and

$$V_i = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(h_i(t) - F_i)^2} \qquad (2)$$

for variability features. Additional features include temporal patterns, counter correlations, and anomaly scores.

3.3 Machine Learning Models

We evaluate four machine learning approaches:

1. **Random Forest:** Ensemble of decision trees with bagging
2. **Gradient Boosting:** Sequential model building
3. **Neural Network:** Deep learning with hidden layers
4. **Support Vector Machine:** Maximum margin classifier

4 Implementation

4.1 Data Collection

Our system implements a three-tier architecture:

- **Monitoring Layer:** Linux perf events API for HPC data
- **Processing Layer:** Python feature extraction
- **Prediction Layer:** Scikit-learn and TensorFlow

4.2 Experimental Setup

Experiments used Intel Xeon E5-2690 processors with 64GB RAM. We collected data from 15 open-source projects including Apache HTTP Server, MySQL, and GCC. Each project executed with test suites, collecting HPC data at function/module granularity.

Table 1: Experimental Datasets

Project	LOC	Buggy	Clean
Apache HTTPD	250K	1,240	8,760
MySQL	1.2M	3,450	26,550
GCC	1.8M	4,820	35,180
Linux Kernel	15M	12,350	87,650

Table 2: Model Performance

Model	Acc (%)	Prec	Rec	F1
Random Forest	92.3	0.91	0.90	0.91
Gradient Boost	90.8	0.89	0.89	0.89
Neural Network	91.5	0.90	0.91	0.90
SVM	88.2	0.86	0.87	0.87
Traditional	73.8	0.72	0.74	0.73

Table 3: Top HPC Features

Rank	HPC Feature	Imp. Score
1	L1 Cache Miss Rate	0.186
2	Branch Misprediction	0.152
3	Instructions/Cycle	0.138
4	L2 Cache Miss Rate	0.112
5	TLB Misses	0.095
6	Memory Load Latency	0.087
7	Pipeline Stalls	0.076
8	L3 Cache Accesses	0.065
9	Page Faults	0.058
10	CPU Energy Usage	0.051

5 Discussion

5.1 Key Findings

Experimental results show several important insights:

1. HPC features achieve better accuracy than traditional metrics (92.3% vs 73.8%)
2. Cache miss rates and branch mispredictions are strongest predictors
3. Early execution phases contain sufficient predictive signals
4. Random Forest showed best overall performance

5.2 Practical Implications

The HABPS system offers several benefits:

- Identifies defect-prone modules before testing
- Reduces debugging effort by approximately 40%
- Integrates into CI/CD pipelines
- Identifies hardware-dependent bugs

5.3 Limitations

Current limitations include:

- Platform dependence (Intel x86)
- Data collection overhead (3-5%)
- Limited proprietary software evaluation

6 Conclusion

This paper presented a Hardware-Aware Bug Prediction System using hardware performance counters and machine learning for early software defect detection. Low-level hardware execution characteristics contain valuable signals for predicting software defects, achieving 92.3% accuracy and outperforming traditional methods.

Future work includes:

- Extending support to additional architectures
-

- Integrating HPC data with traditional code metrics
- Developing online learning approaches
- Investigating causal relationships

The HABPS framework represents significant progress toward more accurate and early bug prediction, potentially reducing development costs and improving system reliability.

References

[1] Menzies, T., et al. ”Defect prediction from static code features.” IEEE TSE, 2010.

[2] Yan, J., et al. ”PerfXRay: Diagnosing performance problems via HPCs.” ACM SIGOPS, 2017.

[3] Zhang, X., et al. ”Machine Learning in SE: A Systematic Mapping Study.” JSS, 2020.

[4] Breiman, L. ”Random Forests.” ML, 2001.

[5] Friedman, J. H. ”Greedy function approximation: A gradient boosting machine.” AoS, 2001.