Research Concept Note: Hardware Behavioral Inference for GPU Performance Optimization

Project: Automated GPU Performance Analysis Using Hardware Performance Counters
Objective: Investigate machine learning approaches for GPU optimization without application instrumentation

Researcher: Andrew Espira
Institution: Saint Peter's University

---

1. Research Problem

Current GPU performance optimization requires application-level instrumentation or manual analysis by domain experts. This creates limitations for legacy applications, proprietary software, and production environments where code modification is not feasible.

Research Question: Can machine learning analysis of hardware performance counters provide actionable GPU performance optimization recommendations without requiring application modification?

2. Technical Approach

2.1 Behavioral Inference Algorithm Framework

Core Hypothesis: Hardware performance counter patterns contain sufficient information to infer application behavior and identify optimization opportunities without application-level instrumentation.

2.2 Data Collection & Feature Engineering

Hardware Performance Counter Collection:

- **GPU Metrics**: SM utilization, memory bandwidth utilization, cache hit/miss rates, power consumption
- **Memory Patterns**: Memory allocation frequency, access stride patterns, coalescing efficiency
- **Temporal Features**: Performance counter time series with microsecond resolution
- **Spatial Features**: Multi-GPU coordination patterns, memory hierarchy utilization

2.3 Machine Learning Pipeline Architecture

Feature Extraction Algorithm:

Classification Algorithms:

- **Random Forest Classifier**: For workload type identification (compute-bound, memory-bound, I/O-bound)
- **DBSCAN Clustering**: For discovering novel performance patterns not in training data
- **Temporal Convolutional Networks**: For time-series pattern recognition in performance degradation
- **Isolation Forest**: For anomaly detection in performance counter distributions

2.4 Privacy-Preserving Analysis Framework

Differential Privacy Implementation

Security Considerations:

- **Side-Channel Attack Mitigation**: Ensure behavioral inference doesn't leak sensitive application data
- **Data Minimization**: Collect only performance-relevant metrics, discard sensitive information
- **Secure Aggregation**: Use federated learning principles for multi-tenant environments

2.5 Optimization Recommendation Engine

Pattern-to-Optimization Mapping Algorithm:

2.6 Real-Time Processing Architecture

Streaming ML Pipeline:

- **Data Ingestion**: Apache Kafka for high-throughput performance counter streams
- **Feature Processing**: Sliding window algorithms for real-time feature extraction
- **Model Inference**: Pre-trained models with <100ms inference latency
- **Recommendation Generation**: Rule-based engine for immediate optimization suggestions

Implementation Stack:

- **eBPF Development**: BCC/libbpf for kernel-level instrumentation
- **ML Framework**: scikit-learn for traditional ML, PyTorch for deep learning components
- **Time Series Processing**: pandas with numba acceleration for performance
- **Privacy Library**: IBM DiffPrivLib for differential privacy implementation
- **Streaming**: Apache Kafka + Redis for real-time data processing

3. Experimental Design

3.1 Phase 1: Data Collection (Weeks 1-3)

- Collect performance data from diverse GPU workloads (ML training, inference, HPC applications)
- Build dataset correlating hardware patterns with manual optimization outcomes
- Validate data quality and feature extraction pipeline

3.2 Phase 2: Algorithm Development (Weeks 4-6)

- Train classification models to identify workload types from hardware signatures
- Develop recommendation algorithms mapping patterns to optimization strategies
- Implement privacy-preserving analysis techniques

3.3 Phase 3: Validation (Weeks 7-8)

- Test system accuracy against expert manual analysis
- Measure performance improvements from generated recommendations
- Document findings and prepare academic submission

4. Expected Outcomes

4.1 Technical Deliverables

- Working system for GPU workload classification from hardware performance counters
- Optimization recommendation engine with measurable accuracy metrics
- Open-source implementation for reproducible research

4.2 Academic Contributions

- Systematic evaluation of hardware performance counters for GPU optimization
- Novel application of behavioral inference techniques to performance engineering
- Framework for privacy-preserving GPU performance analysis

4.3 Success Criteria

- Classification accuracy >70% compared to expert analysis
- Demonstrable performance improvements on benchmark applications
- Reproducible methodology with published datasets

5. Resource Requirements

5.1 Computing Resources

- GPU access via cloud platforms (Google Colab Pro, AWS/GCP credits)
- Storage for performance datasets (estimated 1-5TB)
- Applications submitted for NVIDIA Academic Grant Program

5.2 Technical Infrastructure

- Development tools: Python ecosystem, eBPF toolchain, CUDA toolkit
- Datasets: MLCommons benchmarks, synthetic GPU workloads
- Validation environment: Controlled GPU cluster access

6. Research Significance

6.1 Technical Merit

- Addresses practical limitations of current GPU optimization approaches
- Combines systems programming (eBPF) with machine learning for novel application
- Provides empirical evaluation of hardware-based performance inference

6.2 Broader Impact

- Enables optimization of applications where source code modification is not possible
- Contributes to automated performance engineering research
- Offers privacy-preserving alternative to application-level monitoring

7. Risk Assessment

7.1 Technical Risks

- Hardware performance counters may not provide sufficient optimization signal
- Real-time processing requirements may exceed computational capabilities
- Privacy preservation techniques may reduce optimization effectiveness

7.2 Mitigation Strategies

- Focus on feasibility study rather than production-ready system
- Document limitations and failure cases as valuable research contributions
- Provide fallback to offline analysis if real-time constraints prove unrealistic

8. Questions for Review

1. **Research Scope**: Is the 8-week timeline appropriate for meaningful academic contribution?
2. **Technical Approach**: Are there alternative methodologies that should be considered?
3. **Evaluation Strategy**: What additional validation methods would strengthen the research?

4. **Academic Positioning**: How can this work best contribute to the systems research community?

Contact Information:
Email: masundeespira@gmail.com
Phone: 5518041964
Available for Discussion: 30-minute research consultation