Heterogeneous Non-Linear Performance Modeling of Parameter Interactions

Steven D. Harris
Roger D. Chamberlain
Christopher D. Gill

Steven D. Harris, Roger D. Chamberlain, and Christopher D. Gill, "Heterogeneous Non-Linear Performance Modeling of Parameter Interactions," in *Proc. of IEEE International Parallel and Distributed Processing Symposium Workshops* (IPDPSW), May 2026, pp. 57-66.
DOI: 10.1109/IPDPSW71298.2026.00014

Presented at 28th Workshop on Advances in Parallel and Distributed Computational Models (APDCM), New Orleans, LA, USA.

Dept. of Computer Science and Engineering
Washington University in St. Louis

Heterogeneous Non-Linear Performance Modeling of Parameter Interactions

Steven D. Harris
Roger D. Chamberlain
Christopher D. Gill
Dept. of Computer Science and Engineering
Washington Univ. in St. Louis
St. Louis, MO, USA
{sharris22,roger,cdgill}@wustl.edu

***Abstract*—Important metrics such as execution time and other quantifications of resource usage are not created in isolation. Rather, such metrics pertain to complex non-linear interactions governing behaviors, functions, and operations of myriad parameters. Rather than focusing on a singular metric, we ask: are there hidden insights within metric interactions? Do those interactions hold the keys to new performance insights? If so, where can these parameters be found? What is the method of their identification? What is the process by which the significant parameters may be analyzed?**

Answers to these questions are nuanced and varied. We propose Structural Equation Modeling (SEM) as one relevant tool for finding solutions to assist modern computer engineers in analyzing platforms at the edge. We show that heterogeneous platforms possess a range of performance capabilities with interesting implications for workloads on such platforms. We show that as workloads run on platforms, workload- and platform-specific factors matter, based not only on the number of execution units but their constitution. Consequently, we propose a means of resource usage identification for modern AI/ML workloads, simulated via GEMM intensive operations, revealing that the interaction of workloads and platforms creates signatures that we explore across a selection of big.LITTLE platforms to identify performance implications.

***Index Terms*—asymmetric multiprocessing, ARM big.LITTLE, structural equation modeling**

I. Introduction

This paper explores challenging complexities within performance modeling domains, to show that non-linear performance characteristics and the interactions between platform and workload components are increasingly relevant in heterogeneous platforms. These parameters are necessary to articulate platform performance, and can be used to investigate performance portability of workloads. When considering AI/ML architecture at the edge, homogeneous computing platforms are being superseded rapidly by their heterogeneous counterparts. Though many uniform platforms still exist, the necessary pursuit of general-purpose AI/ML computing has required platform support for ever-increasing numbers of features, protocols, and schema in conventional (e.g., Open Systems Interconnection) layers to make everything from programs and context-based workloads to source-code and user-experiences, generalize in seamless ways across devices. Platforms and workloads must diversify support of heterogeneous resources to maintain both functionality and relevance. Yet, most analysis methods fall short of measuring relationships beyond the linear ones clearly demarcated by upper/lower bounds, and give woefully pessimistic estimates, articulating limited actionable performance prediction or portability.

We contend that measuring performance using un-optimized GEMM workloads across varying dimensions (i.e., $\{M = K = N\} \in \{64, ..., 8192\}$) provides a reasonable proxy for edge-based AI/ML workloads and inference tasks. That abstraction of AI/ML tasks, reducible to well-known mathematical operations, reveals specific bottlenecks including memory bandwidth, cache limits, and compute saturation, ultimately dictating model latency. Some of the main mathematical models employed for AI/ML include:

- Attention Mechanisms (Transformers)
- Dense (Fully Connected) Layers
- Convolutional Layers (CNNs)

We focus on Transformer-based workloads as these are the most relevant to GEMM and are widely used in modern AI/ML. The contributions of this paper include:

- A concrete demonstration of the importance of more robust models than asymptotic ones provide.
- An investigation of how multiple parameters may interact to influence performance.
- The introduction of several new performance indicators, driven from the models developed earlier.

We use structural equation modeling techniques [1], [2] to achieve each of those contributions.

II. Background and Related Work

Modern Transformer architectures integrate three distinct computational primitives, each of which exercises different hardware subsystems while serving specific and critical roles within the cognitive pipeline. The traditional Transformer architecture is a composite of **Dense Position-wise Feed-Forward Network (FFN)** layers and **Attention** mechanisms that are arranged in an interleaved sequence. **Convolutional**

Support provided by NSF grant CNS-2229290 (CPS) and a gift from BECS Technology, Inc.

2995-066X/26/$31.00 ©2026 IEEE
DOI 10.1109/IPDPSW71298.2026.00014

Layers (CNNs), while not part of basic Transformers, support spatial feature extraction and efficiency from hybrid components. The important operations and the hardware they exercise are shown in Table I.

TABLE I
COMPARISON OF AI/ML LAYER TYPES

Layer Type	Operation	Hardware Driver	Cognitive Role
Attention / SDPA	Dynamic Weighting	Memory Bandwidth / Bus	Relationship Modeling
Dense (FFN)	Linear + Non-linear	Compute (TFLOPS) / Cache	Inference / Reasoning
CNN	Sliding Window	Cache Locality / Throughput	Feature Perception

A. SDPA: Ubiquity and Strategic Importance

Scaled Dot-Product Attention (SDPA) has become a fundamental operation of the Transformer architecture. Starting around 2017, SDPA has transitioned from a niche research component to a dominant primitive in modern computing [3]. It also serves as a core engine for Large Language Models (GPT-4, Claude, Llama), Computer Vision (Vision Transformers), and Multimodal systems.

In AI/ML, SDPA is critical because it enables global context modeling. Unlike previous sequential models (RNNs), SDPA allows all tokens in a sequence to interact with each other simultaneously, supporting massive parallelization and enabling capture of long-range dependencies [4]. SDPA is a high-intensity workload that exercises specific hardware tiers (depending on the implementation) as shown in Table II. We next elaborate on the memory movement and complexity of workloads in terms of computation time and latency.

TABLE II
SDPA: RECRUITED HARDWARE RESOURCES

Resource	Role in SDPA	Specific Exercise
GPU	Primary accelerator	Uses **Tensor Cores** for QK^T and PV matrix multiplications
CPU	General execution	High reliance on **AVX-512/AMX** instructions and **L3 cache/DDR5 bus**
ASIC	Specialized logic	Typically uses **Systolic Arrays** for high-throughput matrix operations
FPGA	Reconfigurable logic	Uses custom data paths to avoid bottlenecks for **Softmax** and **LayerNorm**

B. Bottlenecks: The Memory Wall

The primary bottleneck in SDPA is memory-based data movement, as opposed to raw computation power.

- **Memory Bandwidth:** Standard attention generates an intermediate $N \times N$ matrix (e.g., for $N = 8192$, this exceeds 67 million elements). Moving this data between global memory and compute units often exceeds computation time.
- **Arithmetic Intensity:** While matrix multiplication is compute-heavy, other surrounding operations (Scale, Mask, Softmax) have low arithmetic intensity, which can cause compute starvation as the processor to wait for data.
- **Quadratic Complexity:** Memory requirements scale at roughly $O(N^2)$. Doubling sequence length quadruples pressure on both the bus and cache, compounding compute time and exacerbating latency.

C. GEMM as an Appropriate Proxy

A benchmark of unoptimized GEMM with dimensions $M = K = N$ (64 to 8192) serves as a reasonable proxy for SDPA performance for the following reasons.

1) Fundamental Mapping: SDPA is essentially two back-to-back GEMM operations,

- **Stage 1:** $Q \times K^T =$ Scores
- **Stage 2:** Softmax(Scores) $\times V =$ Output

If a system cannot efficiently handle a raw GEMM (e.g., at $M = 8192$), it will fail at high-context attention.

2) Cache/Memory Hierarchy Testing: Incremental scaling (e.g., from 64 to 8192) tracks the performance cliff that need not be the same for a given increment. We hypothesize that multiple edges may exist within the set of increments, for M in the following ranges:

- **Small M (64–256):** Fits in L1/L2 cache. Reflects raw instruction throughput.
- **Medium M (512–2048):** Spills to L3 cache. Reflects cache latency.
- **Large M (4096–8192):** Exceeds on-chip cache, forcing reliance on RAM/HBM bandwidth.

A comprehensive break down of the anticipated performance inflections across the performance region (e.g., for M from 64 to 8192) is given in Table III.

TABLE III
ARCHITECTURAL MAPPING AND HARDWARE BOTTLENECKS FOR SBCS

Scale	Matrix Size	Inference Proxy	Primary Bottleneck
Small	64×64 128×128 256×256	Elementwise / Pointwise Conv Vectorized Math (NEON/SIMD) Small GEMM Blocks	Kernel Launch Overhead Instruction-Level Parallelism L1/L2 Cache Speed
Medium	512×512 1024×1024	Tiled Convolutions Quantized Dense Layers	L2 Cache Reuse / Latency Main Memory Bandwidth
Large	2048×2048 >2048×2048	Edge Model Hidden Layers Stress-Case Workloads	Bandwidth / Cache Thrashing RAM Capacity / Thermals

3) Prediction of Arithmetic Intensity: Unoptimized GEMM workloads reveal the base of the model, or the hardware's "ground-floor." Because optimized kernels use tiling to mimic small GEMM behavior, knowing raw performance across these scales allows architects to predict potential speedup from software optimization versus a strict hardware limitation with granularities in the following scales.

- **Small Scales (64×64 to 256×256)**: These dimensions often fit entirely within the L1 or L2 cache. Performance here reflects the raw clock speed and instruction-level parallelism of the CPU or NPU. High latency at this scale indicates poor compiler optimization or high overhead in the inference engine (e.g., TensorFlow Lite or ONNX Runtime).
- **Medium Scales (512×512 to 2048×2048)**: These typically exceed the fast cache and require constant data movement from SRAM or DDR memory. This is the "sweet spot" for most edge vision models (like MobileNet [5]). Latency here is a direct proxy for the device's memory bandwidth.
- **Large Scales (4096×4096 to 8192×8192)**: These dimensions represent the large weight matrices of LLMs or high-resolution transformers. On resource-limited edge devices, these sizes trigger thermal throttling and page swapping (if matrices exceed RAM). Execution time at this scale predicts whether the device will crash or experience "stuttering" during long inference tasks.

The typical AI/ML workload is stochastic and involves saving intermediate "activations" for the backward pass, which consumes significant memory. In inference, the weights are static (i.e., read-only). Thus, the execution time of a static GEMM operation is almost perfectly correlated with the time it takes to process one "chunk" of data through a model layer. If a device takes T milliseconds to multiply a matrix (e.g., $N \geq 8192$), we can calculate the Tokens Per Second (TPS) or Frames Per Second (FPS) throughput of a model with a known parameter count, as the compute-to-memory ratio remains constant for that specific hardware-matrix pairing.

D. Related Work

Analytical Performance Models: Traditional performance modeling relies on asymptotic analysis for the PRAM model [6] and the Roofline Model [7]. While the Roofline Model identifies bounds based on operational intensity and peak bandwidth, it often fails to capture the non-linear performance "valleys" and "plateaus" inherent in heterogeneous edge hardware. Existing research into execution time models [8] for General Matrix-Matrix Multiplication (GEMM) frequently assumes linear scaling with respect to FLOPs, overlooking the stochastic nature of memory contention and cache misses in non-uniform memory access (NUMA) architectures.

Heterogeneous Edge Intelligence: Current literature on AI/ML at the edge focuses on hardware-specific optimizations for Transformers and CNNs [9]. Research into performance Portability (e.g., SYCL [10], Kokkos [11]) attempts to abstract hardware layers, yet performance prediction remains difficult due to the "black-box" nature of proprietary drivers and varying compute-to-memory ratios [11]. Recent studies have identified that Transformer inference latency is dominated by GEMM operations, specifically within the self-attention mechanism, necessitating high-fidelity proxies that go beyond simple throughput metrics [12].

Structural Equation Modeling (SEM) in Computing: SEM is traditionally utilized in social sciences to model complex relationships between observed and latent variables [1]. Its application to computer systems is a nascent but growing field [13]. Unlike standard regression, SEM allows for the simultaneous evaluation of multi-variate dependencies, making it uniquely suited for capturing the interactions between workload dimensions and architectural bottlenecks like compute saturation and interconnect latency.

We investigate the limitations of existing performance models in the context of heterogeneous edge computing. We argue that traditional asymptotic analysis is insufficient for predicting performance of modern AI/ML workloads, which have complex, non-linear behaviors due to hardware-software interactions.

III. Scope

Modern platforms have myriad capabilities, but the most challenging and resource intensive operations at the edge include both artificial intelligence (AI) and machine learning (ML) workloads. Critical edge-platforms need to offer high-performance, specifically for un-optimized, 3-loop, general matrix multiplication (GEMM) whether through highly optimized source code, or architectural levers (specialized instructions, advantaged pipelines, data-specific accelerators, operations-in-circuits). Efficacy of the platform in accommodating such workloads is essential for real-world tasks (e.g., AI sensing, decision processes, and other critical contextual scenarios) which depend on the support of entire classes of edge-based AI/ML powered devices. Therefore, it is both appropriate and pertinent to leverage the ability of these platforms to execute GEMM-based calculations as a relevant and timely measurement in assessing platforms capabilities for edge-based AI/ML workloads. For sake of discussion and to scope this paper, we restrict our attention to ARM big.LITTLE based single board computers. These platforms, with respect to processor core architecture, have enough internal heterogeneity (high-power/low-efficiency and low-power/high-efficiency cores) and external similarity (Big high-performance and Little high-efficiency islands) to offer a valid interpretation of modern heterogeneous architecture that requires available and sufficient analysis methods to inform platform performance articulation for critical AI/ML workloads.

A. Orange Pi 5 Configuration

The Orange Pi 5 offers a selection of high-performance ("Big") cores and energy-efficient ("Little") cores, each with unique performance implications. The platform utilizes a Rockchip RK3588S big.LITTLE architecture which consists of physical (Big & Little) islands, each with private L1/L2 caches, and one logical (Dual) island composed of at least one core from *each* physical island. All cores are connected via the last-level cache. The most pertinent performance variables are listed in Table IV, offering a plethora of configuration options.

TABLE IV
ORANGE PI 5 SPECIFICATIONS

Variables	Performance Components
Number of Cores	ARM Quad-core A76 + Quad-core A55
Island Selection	High-Performance (Big) + Energy-Efficient (Little)
Core Frequencies	2.4 GHz down to 1.8 GHz
Power Levels	Conservative, Power Save, Performance

B. Radxa Cubie A7A Configuration

The Radxa Cubie A7A utilizes an Allwinner A733 big.LITTLE architecture. The Cubie A7A has a heterogeneous octa-core configuration consisting of two distinct physical islands: one performance-optimized cluster of two ARM Cortex-A76 ("Big") cores and one power-optimized cluster of six ARM Cortex-A55 ("Little") cores. The clusters are integrated by a unified interconnect with shared access to high-bandwidth LPDDR5 memory. The most pertinent performance variables are listed in Table V, offering many configuration options.

TABLE V
RADXA CUBIE A7A SPECIFICATIONS

Variables	Performance Components
Number of Cores	ARM Dual-core A76 + Hexa-core A55
Island Selection	High-Performance (Big) + Energy-Efficient (Little)
Core Frequencies	2.4 GHz down to 0.4 GHz
Power Levels	Conservative, Power Save, Performance

C. General Matrix Multiply

To exercise this platform, we selected matrix-matrix computations of the form: $C \leftarrow \gamma AB + \delta D$, where $\gamma, \delta \in \mathbb{R}$, $C, D \in \mathbb{R}^{M \times N}$, $A \in \mathbb{R}^{M \times K}$, and $B \in \mathbb{R}^{K \times N}$. The workload, authored in C++, is a common OpenMP parallelized "three-loop" naïve matrix multiply implementation. It does not make use of any optimizations that leverage prior hardware knowledge. This workload reflects a common general matrix multiply (GEMM) computation, parallelized with OpenMP for threads in the range $\{1, ..., 2P\}$, where P is the number of processor cores. The dimensions ($M = K = N$) of the square matrices are in the range $\{64, 128, ..., 8192\}$, which is the $\mathtt{M}_c$ attribute. This workload is representative of computationally intensive requests that are essential and common for many edge-based applications, in particular signal processing (a historical task on the edge) and machine learning (a newly emerging task on the edge). We collected measurements using `perf` for each workload configuration.

D. Top-Down Approach Rationale

For this investigation, we performed a parameter reduction by fixing parameters in the upper layers (e.g., 1 through 4) as illustrated in Table VI and varying only the physical hardware options. By alternating the platforms we can identify and articulate performance for classes of edge-based platforms (i.e., single-board computers). Specifically, we are interested in hardware platforms utilizing heterogeneous processors (i.e., big.LITTLE) that support mission-critical AI/ML-based workloads (e.g., GEMM). Consequently, we look at how the hardware options affect performance even for similarly equipped devices (Orange Pi 5 + Cubie A7A).

TABLE VI
PLATFORM LAYERS AND PARAMETERS

Layer	Key Parameters
1 Execution	Resource Count: {Islands, Cores, ...}
2 Scheduling	Parallelization Scale: {Threads, Replications, ...}
3 Workload	Problem Size: {Matrix Dimension, Elements, ...}
4 Platform	Parameter: $\{L_1, L_2, L_3$, Context Switches, ...$\}$
5 Hardware	Execution Unit: $\{Orange, Cubie\} \times \{Big, Little\}$

We use a top-down approach that fixes most of the parameters in layers 1 through 4 and selects collections of objects by alternating between the device options available in the physical hardware layer as shown in Table VI. For reference, the elements in layer 5 are written in Cartesian product notation as follows:
`Platform` = {Orange, Cubie}
`Island` = {Big, Little}
`Execution Unit` = `Platform` × `Island` = {(Orange, Big), (Cubie, Big), (Orange, Little), (Cubie, Little)}

By iterating through the hardware options, in a rigorous brute-force manner, we aim to reveal performance and workload relationships and behavior at key granularities across these platforms.

IV. POLYNOMIAL PERFORMANCE MODELING & ANALYSIS

To step through the many configuration options associated with our workload, we require a starting point from which to direct the assessment of platform performance. Yet, given the similarly equipped octa-core platforms (i.e., Orange Pi 5 and Cubie A7A) how can we assess the performance capabilities of these devices for any workload let alone our unique one? This investigation then requires us to consider several questions:

1) What are the performance implications for GEMM?
2) What is the model for GEMM?
3) Are we interested in a single parameter or all the parameters?

A. GEMM Modeling and Analysis

We know that the standard 3-loop GEMM computation runs in $\Theta(M^3)$ time where $M = N = K$. As this has the strongest impact on runtime performance, we traditionally model performance based solely on this term. If all machines had exactly the same performance for a GEMM workload, this notion of modeling with the single term makes sense as the result would be both certian and absolute across platforms. Yet, we know this to be untrue, as different devices have different architectural and/or programmatic advantages enabling different terms to vary from platform to platform. Other terms may be required in the consideration of GEMM performance on a platform, but we must consider some regressions of parallel execution time on matrix dimension ($\mathtt{parallel_time} \sim M_i^j$) to understand their importance.

B. Polynomial Modeling

This section details our experimental process and results for modeling the relationship between the predictor variable, `matrix_dimension` (M), and the outcome variable, `parallel_time` (Y), using polynomial regression (e.g., $Y \sim f(M) + \epsilon$). The primary objective is to identify non-linearity and determine if it is significant enough to warrant additional terms for accurate functional model representation.

We start with an initial model which predicts the outcome variable `parallel_time` based on a centered linear term ($\mathtt{M}_c$), a squared term ($\mathtt{M}_c^2$), and a cubed term ($\mathtt{M}_c^3$) of a predictor variable ($\mathtt{M}_c$). The model formula is as follows:

$$\text{parallel_time} \sim \beta_0 + \beta_1 \mathtt{M}_c + \beta_2 \mathtt{M}_c^2 + \beta_3 \mathtt{M}_c^3 + \epsilon.$$

Our investigation will be to determine which terms in the polynomial are significant and should be included for the purposes of modeling.

1) Experimental Steps and Rationale: The analysis follows a four-step methodology, progressing from model preprocessing to formal statistical comparison in order to identify the significant polynomial terms. Those steps are as follows:

(a) **Variable Creation and Centering.** To capture the features of the data, polynomial regression mandates **centering** the predictor variable ($\mathtt{M}_c = M - \bar{M}$) prior to creating polynomial terms ($\mathtt{M}_c^2, \mathtt{M}_c^3$) This step is used to mitigate **multicolinearity** between the linear higher-order terms, thereby ensuring more stable and interpretable regression coefficients.

(b) **Full Model Estimation.** The regression models in our analysis use different combinations of matrix dimension terms to assess the individual significance of the proposed polynomial effects (linear, quadratic, and cubic). Using those findings we are able to identify significant terms critical for performance modeling, not only for the Orange Pi but also for the Cubie A7A platform.

(c) **Model Visualization.** A diagnostic plot was generated for the datasets with the parameters $\mathtt{M}_c$ (centered) and `parallel_time` (scaled). The plot is overlayed with a cubic fitted curve onto the raw data points as shown in Figure 1 and Figure 2 for the two platforms. This provides visual confirmation of the non-linear relationship suggested by the significant coefficients. The visualizations confirm the complex, non-linear curvature accurately captured by the polynomial terms.

(d) **Formal Model Comparison.** Finally, to justify the inclusion of any terms, a nested model comparison was performed using linear model (lm) regression and the analysis of variance (ANOVA) on the data. We used the model fit Information Criteria (IC) to discover the full model. Consequently, the results of the *AIC/BIC* (see description below) indicate that the full cubic model exhibited the **lowest AIC and BIC** values, favoring the most complex specification.

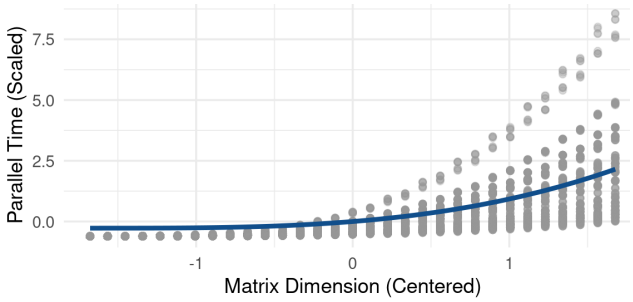


Fig. 1. Orange Pi Model Visualization

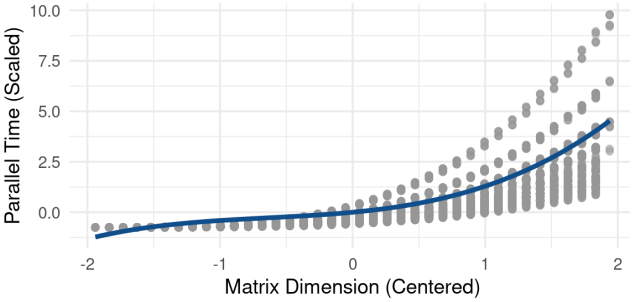


Fig. 2. Cubie Model Visualization

2) Akaike and Bayesian Information Criteria: AIC & BIC: Information criteria are statistical tools used for **model selection**. They provide a quantitative means to compare different mathematical models applied to the same dataset, balancing the accuracy of the fit against the complexity of the model [14].

The *AIC* formula is:

$$AIC = 2k - 2\ln(\hat{L})$$

where k is the number of estimated parameters (complexity) and $\hat{L}$ is the maximum likelihood of the model (goodness of fit). The *AIC* estimates the relative amount of information lost by a given model. The goal is to select a model that describes the data well without unnecessary complexity.

The **BIC**, derived from Bayesian probability, serves a similar purpose to *AIC* but introduces a stronger penalty for models with more parameters, especially as the sample size increases. The *BIC* formula is:

$$BIC = k\ln(n) - 2\ln(\hat{L})$$

where n is the number of observations (sample size) and $k\ln(n)$ is the complexity penalty, which scales with data volume.

The "lower is better" rule is a direct result of the mathematical structure of these equations, which function as loss functions. Both expressions contain the term $-2\ln(\hat{L})$.

- Likelihood ($\hat{L}$) measures how well the model explains the observed data. A "perfect" fit has a high likelihood.
- The negative sign means that as the model's fit improves (higher $\hat{L}$), the overall score **decreases**.

Adding more variables (increasing k) will almost always improve the fit ($\hat{L}$), but it risks **overfitting** which captures noise rather than the underlying signal.

- The terms *AIC* and *BIC* act as a "cost" or "tax" on complexity.
- To achieve a lower total score, a model must improve the fit significantly enough to outweigh the mathematical cost of adding new parameters.

A lower *AIC* or *BIC* indicates a model that achieves the highest possible likelihood with the fewest possible parameters.

3) Data Preparation and Models Tested: The independent variable, matrix dimension ($\mathtt{M}_c$), was centered, and polynomial terms were generated ($\mathtt{M}_c^2$, $\mathtt{M}_c^3$). The dependent variable is `parallel_time`.

The models are shown in Table VII.

TABLE VII
TESTED POLYNOMIAL REGRESSION MODELS

Model	Description	Equation
1	Linear	$\texttt{parallel_time} \sim \mathtt{M}_c$
2	Quadratic	$\texttt{parallel_time} \sim \mathtt{M}_c + \mathtt{M}_c^2$
3	Cubic	$\texttt{parallel_time} \sim \mathtt{M}_c + \mathtt{M}_c^2 + \mathtt{M}_c^3$

All model comparisons relied on the standard *lm* function for reliable *ANOVA* tests. The information criteria (*AIC* and *BIC*) provide a measure of relative fit, where **lower values indicate a better model**. Those results are given in Table VIII, with the estimates for the regression parameters given in Table IX. The results show clearly that all three predictors are **highly significant**.

TABLE VIII
MODEL FIT INFORMATION CRITERIA COMPARISON: ORANGE PI 5 VS. CUBIE A7A

		Orange Pi 5		**Cubie A7A**	
Model	**Description**	*AIC*	*BIC*	*AIC*	*BIC*
1	Linear	17889	17903	80757	80774
2	Linear + Quadratic	16589	16610	55114	55140
3	Linear + Quadratic + Cubic	**16541**	**16569**	**53266**	**53301**

Given that Model 3 has been determined to be the most informative for the prediction of parallel execution time of the workload on both platforms, we decided to look at the model regression to understand the statistical significance and importance of the parameters. The results of the full model regression are shown in Table IX.

C. Linear and Non-Linear Effects

The results indicate that all of the matrix terms are significant which warrants their inclusion in the full model. Given that the standardized estimate coefficients are unitless and all variables have been converted to a standard deviation scale (mean = 0, standard deviation = 1), we can determine the strength and magnitude of the change in the dependent variable by the value of the coefficient, measured in its own standard deviations, assuming all other predictor variables are held constant. This can be used primarily for comparing the relative strength of different paths within the same model. Because the coefficients are unitless, a direct identification of the predictor with the strongest influence on an outcome variable can be made.

The parameters represent both linear and non-linear effects that have a statistically significant impact on parallel execution time. Starting with the *Orange Pi 5*, we outline the parameters and their effects on the outcome variable (parallel execution time) as follows:

- **Linear Effect:** The centered matrix dimension term ($\mathtt{M}_c$) is statistically significant ($p < 0.001$, Estimate $= 0.521$). This indicates a strong, positive **linear** association: as the centered matrix dimension increases, parallel time increases, independent of the non-linear curvature.
- **Non-Linear Effects:**
  - The quadratic term ($\mathtt{M}_c^2$) is highly significant ($p < 0.001$, Estimate $= 0.334$). This indicates a significant non-linear effect where the rate of increase in parallel time accelerates as the matrix dimension grows.
  - The cubic term ($\mathtt{M}_c^3$) is also highly significant ($p < 0.001$, Estimate $= 0.072$). This indicates a more complex, S-shaped curvature or an inflection point is necessary to fully model the relationship.

The relationship between matrix dimension and parallel time is unquestionably non-linear, and all three polynomial terms contribute significantly to predicting the outcome.

V. MULTI-PARAMETER INTERACTION MODELING & ANALYSIS

We next focus on models that incorporate the interaction between multiple parameters. To quantify the performance overhead and efficiency gains using statistical modeling of heterogeneous computing architectures (ARM big.LITTLE), we take the standard approach of assigning the Big cores as the *dummy variable* (or reference group) while explicitly modeling the Little core parameter in equation models to support enhanced measurements.

Modeling with the Little island as the active (dependent) variable allows for direct measurement of power-to-performance scaling. The Little cores being designed for deterministic, low-power background tasks, have coefficients that determine the "threshold of utility:" that is, the point at which the task becomes too intensive or too expensive for the Little island and requires preemptive migration to the Big island.

A. Interaction Effects and Thermal Throttling

Given that we are exploring new models with unknown features, interactions, and parameters, under this scenario the Little cores are often the variable of interest given that their performance is more sensitive to resource contention and

TABLE IX
COMPARATIVE REGRESSION AND VARIANCE RESULTS: ESTIMATE, STANDARD ERROR, AND SIGNIFICANCE

	Estimate		Std. Error		$P(>\|z\|)$
Variable	**ORANGE PI 5**	**Cubie A7A**	**ORANGE PI 5**	**Cubie A7A**	**All Models**
$\mathtt{M}_c$	0.521	0.623	0.020	0.005	0.000
$\mathtt{M}_c^2$	0.334	0.401	0.009	0.002	0.000
$\mathtt{M}_c^3$	0.072	0.094	0.010	0.002	0.000
parallel_time	0.486	0.207	0.008	0.001	0.000

shared cache (L2/L3) constraints. Models based on the Little island can be used to track:

- **Context Switching Overhead:** Determines the cost of moving a thread from the reference (Big) to the modeled (Little) core.
- **DVFS (Dynamic Voltage and Frequency Scaling) Sensitivity:** Little cores often have narrower frequency bands. Modeling them explicitly captures the precise saturation rate compared to the Big core baseline.

B. Mathematical Identification

Using the Big cores as the reference can simplify speedup factor interpretation. When the variable of interest is a metric like parallel execution time (parallel_time), then a positive coefficient for Little-based measurements will immediately identify increased latency (performance degradation) as opposed to a negative coefficient indicative of decreased latency (performance improvement).

C. Deeper Dive – Additional Parameters

TABLE X
NOTATION AND DESCRIPTION

Parameter	Description
$\mathtt{M}_c$	Matrix Dimension Centered
$\mathtt{M}_c^2$	Matrix Dimension Centered Squared
$\mathtt{M}_c^3$	Matrix Dimension Centered Cubic
island	Compute Island *(Baseline: Big Island)*
cores	Compute Unit *(Baseline: 1 Core)*
$\mathtt{M}_c \times$ island	Linear **x** Island *(Interaction)*
$\mathtt{M}_c^2 \times$ island	Quadratic **x** Island
$\mathtt{M}_c^3 \times$ island	Cubic **x** Island
$\mathtt{M}_c \times$ cores	Linear **x** Core
$\mathtt{M}_c^2 \times$ cores	Quadratic **x** Core
$\mathtt{M}_c^3 \times$ cores	Cubic **x** Core
$\mathtt{M}_c \times$ island $\times$ cores	Linear **x** Island **x** Core
$\mathtt{M}_c^2 \times$ island $\times$ cores	Quadratic **x** Island **x** Cores
$\mathtt{M}_c^3 \times$ island $\times$ cores	Cubic **x** Island **x** Core

With the parameters fully identified for the GEMM workload operation, we next explore the individual impacts of the most prominent and obvious parameters common to both platforms: *matrix dimension*, *cores*, and *island*. Using the additional parameters, we derive our first SEM based model, with the notation given in Table X, and the parameter interactions denoted with an '**x**' between parameters. The model depicts the relationship between the independent variables (platform parameters) and the dependent variable (parallel_time).

This SEM path model enables the rapid interpretation of parameter effects. The estimates given in Table XI are the standardized path coefficients, also known as beta coefficients (e.g., β_i). For example, the coefficient for $\mathtt{M}_c$ (i.e., 0.940) indicates that a one-standard-deviation increase in this exogenous (independent) variable corresponds to a 0.940 standard-deviation *increase* of the endogenous (dependent) variable parallel_time. The cores parameters follows a similar pattern but has a marked *decrease* in parallel execution time. The signs of the estimates are indicative of the direction of change with respect to the parallel execution time.

This presentation modality allows the parameter effects to be read easily from the model estimates, to understand the endogenous variables impacts directly. Other interesting insights gained from the parameter estimates can be seen with the values for both $\mathtt{island}_L$ (0.137) and its interaction with matrix terms (e.g., $\mathtt{M}_c \times \mathtt{island}_L$) which all *increase* parallel execution time. The behavior has been determined numerically and via rigorous statistical analysis as opposed to direct architectural insights about the platform known prior to analysis. This is a potent and powerful tool as it can rightfully identify parameters and their contributions (whether positive or negative) to platform performance.

D. Orange Pi 5 Conclusions

Notice that the final model, which included the common parameters (cores, island) along with their interaction with the matrix terms has rounded out the model to an excellent $R^2 = 0.937$. We have identified the highly significant parameters that explain the parallel execution time in our model. Interestingly enough, the difference between Model 3 ($R^2 = 0.514$) based on 3 parameters and Model 4 ($R^2 = 0.937$) based on 23 parameters increased the proportion of explained variance by over 80%.

Given that the standardized coefficients are unit-less and all variables have been converted to a standard deviation scale (mean = 0, standard deviation = 1), we can determine the strength and magnitude of the change in the dependent variable by the value of the coefficient, measured in its own standard deviations, assuming all other predictor variables are held

constant. This can be used primarily for comparing the relative strength of different paths within the same model. Because the coefficients are unitless, you can compare directly which predictor has the strongest influence on an outcome variable.

Given that the models are explaining execution time, in terms of the metric parallel_time, these estimates are able to bring a cohesion of both direction and magnitude to performance modeling components in computer engineering. Considering the three matrix parameters, while the linear component ($\mathtt{M}_c$ = 0.940) has the largest magnitude (and the most impact) of the terms, the secondary ($\mathtt{M}_c^2$ = 0.511) and tertiary ($\mathtt{M}_c^3$ = 0.208) terms also have a significant impact. Simultaneously, the sign of the estimates informs us of the direction of impact. In this instance the estimate for the matrix dimension terms has increased as compared to the matrix-term-only models (Models 1-3). The full model for the Orange Pi 5 is given in Table XI.

E. Additional Execution Platform

We found that the matrix model for the Orange Pi 5 only accounted for an R^2 of 0.51 where as the full model accounted for an R^2 of 0.937 . This suggests that additional parameters improve prediction of the parallel execution time across the various configurations. Given the value of this model, we determined to run it with the same parameters for the Cubie A7A and found a similiar high-significant model. Results for the Cubie A7A are given in Table XII.

TABLE XI
ORANGE PI - MODEL REGRESSIONS AND VARIANCES – $R^2 = 0.937$

	Est.	Std.Err	$P(>\|z\|)$
Regressions:			
parallel_time ∼			
$\mathtt{M}_c$	0.940	0.028	0.000
$\mathtt{M}_c^2$	0.511	0.011	0.000
$\mathtt{M}_c^3$	0.208	0.028	0.000
$\mathtt{island}_L$	0.137	0.005	0.000
$\mathtt{cores}_2$	−0.150	0.009	0.000
$\mathtt{cores}_3$	−0.190	0.009	0.000
$\mathtt{cores}_4$	−0.167	0.007	0.000
$\mathtt{M}_c \times \mathtt{island}_L$	0.350	0.012	0.000
$\mathtt{M}_c^2 \times \mathtt{island}_L$	0.275	0.006	0.000
$\mathtt{M}_c^3 \times \mathtt{island}_L$	0.110	0.012	0.000
$\mathtt{M}_c \times \mathtt{cores}_2$	−0.358	0.018	0.000
$\mathtt{M}_c \times \mathtt{cores}_3$	−0.491	0.019	0.000
$\mathtt{M}_c \times \mathtt{cores}_4$	−0.358	0.013	0.000
$\mathtt{M}_c^2 \times \mathtt{cores}_2$	−0.268	0.010	0.000
$\mathtt{M}_c^2 \times \mathtt{cores}_3$	−0.349	0.010	0.000
$\mathtt{M}_c^2 \times \mathtt{cores}_4$	−0.278	0.008	0.000
$\mathtt{M}_c^3 \times \mathtt{cores}_2$	−0.080	0.019	0.000
$\mathtt{M}_c^3 \times \mathtt{cores}_3$	−0.106	0.019	0.000
$\mathtt{M}_c^3 \times \mathtt{cores}_4$	−0.079	0.013	0.000
$\mathtt{island}_L \times \mathtt{cores}_2$	0.029	0.006	0.000
$\mathtt{M}_c \times \mathtt{island}_L \times \mathtt{cores}_2$	0.061	0.012	0.000
$\mathtt{M}_c^2 \times \mathtt{island}_L \times \mathtt{cores}_2$	0.053	0.007	0.000
$\mathbf{M}_c^3 \times \mathbf{island}_L \times \mathbf{cores}_2$	0.018	0.012	**0.115**
Variances:			
parallel_time	0.063	0.001	0.000

TABLE XII
CUBIE A7A - MODEL REGRESSIONS AND VARIANCES ($R^2 = 0.950$)

	Est.	Std.Err	$P(>\|z\|)$
Regressions:			
parallel_time ∼			
$\mathtt{M}_c$	0.674	0.036	0.000
$\mathtt{M}_c^2$	0.442	0.015	0.000
$\mathtt{M}_c^3$	0.206	0.035	0.000
$\mathtt{island}_L$	0.070	0.003	0.000
$\mathtt{cores}_2$	−0.065	0.010	0.000
$\mathtt{cores}_3$	−0.311	0.006	0.000
$\mathtt{cores}_4$	−0.366	0.006	0.000
$\mathtt{M}_c \times \mathtt{island}_L$	1.111	0.042	0.000
$\mathtt{M}_c^3 \times \mathtt{island}_L$	0.353	0.040	0.000
$\mathtt{M}_c \times \mathtt{cores}_2$	−0.117	0.019	0.000
$\mathtt{M}_c \times \mathtt{cores}_3$	−0.657	0.013	0.000
$\mathtt{M}_c \times \mathtt{cores}_4$	−0.837	0.015	0.000
$\mathtt{M}_c^2 \times \mathtt{cores}_2$	−0.096	0.010	0.000
$\mathtt{M}_c^2 \times \mathtt{cores}_3$	−0.527	0.007	0.000
$\mathtt{M}_c^2 \times \mathtt{cores}_4$	−0.659	0.007	0.000
$\mathbf{M}_c^3 \times \mathbf{cores}_2$	−0.032	0.018	**0.079**
$\mathtt{M}_c^3 \times \mathtt{cores}_3$	−0.195	0.013	0.000
$\mathtt{M}_c^3 \times \mathtt{cores}_4$	−0.253	0.014	0.000
$\mathtt{island}_L \times \mathtt{cores}_2$	−0.101	0.011	0.000
$\mathtt{M}_c \times \mathtt{island}_L \times \mathtt{cores}_2$	−0.185	0.020	0.000
$\mathtt{M}_c^2 \times \mathtt{island}_L \times \mathtt{cores}_2$	−0.160	0.011	0.000
$\mathbf{M}_c^3 \times \mathbf{island}_L \times \mathbf{cores}_2$	−0.057	0.019	**0.003**
Variances:			
parallel_time	0.050	0.000	0.000

F. Discussion

This study utilizes structural equation modeling to deconstruct performance variance in General Matrix Multiplication (GEMM) across two distinct ARMv8 heterogeneous architectures: a balanced 4+4 big.LITTLE configuration (Orange Pi 5) and an asymmetric 2+6 configuration (Cubie A7A). By regressing execution time against polynomial workload complexity ($\mathtt{M}_c, \mathtt{M}_c^2, \mathtt{M}_c^3$) and core-island interactions, we identify a critical *asymmetry inversion point*. While the 2+6 topology offers superior area-efficiency for low-order workloads, it exhibits a statistically significant "congestion collapse" in high-order cubic scaling ($p < 0.01$). We demonstrate that the $\mathtt{M}_c^3 \times \mathtt{island}_L \times \mathtt{cores}_P$ interaction term serves as a proxy for cache-coherency traffic and memory bus contention, revealing that the 6-core Little cluster in the Cubie A7A suffers from an accelerated transition to memory-boundedness compared to the 4-core cluster of the Orange Pi 5.

Heterogeneous System-on-Chip (SoC) designs in our analysis have bifurcated into two primary philosophies: balanced performance (e.g., Rockchip RK3588's 4+4) and throughput-optimized asymmetry (e.g., Allwinner A733 2+6 configurations). While a 2+6 topology maximizes multi-threaded efficiency for background tasks, the behavior under high-intensity arithmetic kernels like GEMM remains under-theorized.

1) The Stochastic Inversion Point: One key finding is the significance of the triple interaction term $\mathtt{M}_c^3 \times \mathtt{island}_L \times \mathtt{cores}_2$.

- **Orange Pi 5 ($p = 0.115$):** The insignificance suggests a "symmetric memory-compute ratio" where complexity scales independently of core count on the Little island.
- **Cubie A7A ($p = 0.003$):** The high significance identifies a "non-linear coupling." On the Cubie, as matrix size and core count increase, performance degrades stochastically as a function of cluster contention rather than linearly.

Consequently, asymmetry breeds non-determinism. The 4+4 setup allows for decoupled scheduling heuristics, whereas the asymmetric 2+6 architecture requires a multivariate-aware scheduler.

2) Island Penalty and Scaling Sensitivity: The parameter $\texttt{island}_L$ represents the base latency penalty for offloading tasks to the energy-efficient cores. With respect to the Cubie A7A, the base estimate is lower (0.070) than the Orange Pi 5 (0.137). However, the interaction term $\texttt{M}_c \times \texttt{island}_L$ reveals a strong divergence:

- **Orange Pi 5:** 0.350
- **Cubie A7A:** 1.111

The $3.17\times$ higher sensitivity in the Cubie A7A indicates that the 6-core Little island hits a structural performance wall. This is a manifestation of *architectural divergence*, where the 6-core cluster likely shares a narrow path to the L3 cache, creating a "funnel effect."

VI. New Performance Indicators

In this section, we define several new metrics that are derived from the SEM estimates articulated above. Their values are summarized in Table XIII and each is discussed in turn.

A. Complexity Sensitivity ($\sum \beta_i \texttt{M}_c^j$)

This measures the system's performance volatility as task complexity increases.

- **Variables:** β_i represents the standardized path coefficient (weights), and $\texttt{M}_c^j$ is the complexity level of task j.
- **Analysis:** The Orange Pi 5 (1.659) is more sensitive to complexity than the Cubie (1.322). While more powerful, the Orange Pi's performance fluctuates more based on the specific type of instruction set being processed.

B. Inter-Island Scaling Friction ($\texttt{M}_c \times I_L$)

This calculates the overhead penalty incurred when data travels between different core clusters (e.g., big.LITTLE architecture).

- **Variables:** $\texttt{M}_c$ is the computational load; I_L is the inter-island latency.
- **Analysis:** The Cubie A7A (1.111) experiences over $3.17\times$ more friction than the Orange Pi (0.350). This indicates a significant communication bottleneck in the Cubie's internal bus or interconnect.

C. Multi-Core Saturation ($\Delta\beta_i C_3 \to C_4$)

This reflects the marginal utility of adding a fourth core to a parallel task.

- **Variables:** $\Delta\beta$ is the change in the performance coefficient during the transition from 3rd to 4th core ($C_3 \to C_4$).
- **Analysis:** The Orange Pi displays **saturation** ($+0.023$), given that the overhead of managing an additional 4th core may actually degrades performance due to competition for resources, whereas the Cubie remains efficient (-0.055).

D. Stochastic Coupling Significance (p value)

A statistical check determining if the relationship between architectural "noise" and performance is reliable.

- **Analysis:** The Orange Pi ($p = 0.115$) is not statistically significant, meaning its performance is relatively independent of random noise. The Cubie ($p = 0.003$) is highly significant, suggesting its performance is tightly (and potentially negatively) coupled with architectural fluctuations.

E. Critical Complexity Index (χ)

We hypothesize that adding more Little cores (6 vs. 4) may paradoxically reduce system efficiency particularly for high-matrix workloads due to coherency traffic overwhelming the shared interconnect. We propose a *critical complexity index* (χ) to validate this hypothesis:

$$\chi = \frac{\sum |\beta_{\text{Interaction}}|}{\sum |\beta_{\text{Direct}}|} \tag{1}$$

- **Variables:** χ enables a scheduler to identify the point where the overhead of the Little island exceeds its computational contribution.
- **Analysis:** A $\chi > 2.0$ suggests "interaction bound" rather than "compute Bound" architecture for complex workloads ($\texttt{M}_c$). Notice the operation threshold, particularly, when χ significantly exceeds 1.0, the Little cores ($\texttt{island}_L$) may act as net liabilities for complex tasks. On the Cubie A7A, the high χ dictates that the scheduler should force-idle the 6 Little cores for large $\texttt{M}_c$ to prevent "poisoning" the memory bus for the 2 Big cores. The *efficiency implications* of the high χ on Cubie validates our hypothesis.

The interaction overhead as measured by this new metric, the critical complexity index χ, is as follows:

$$\chi_{Cubie} \approx \mathbf{2.85} \qquad \chi_{OrangePi} \approx \mathbf{1.42}$$

The χ_{Cubie} magnitude for interaction overhead is over double its direct computational contribution compared to the Orange Pi. This indicates extreme architectural fragility where coordination costs dominate execution time. The approximate $2\times$ difference in the ratio disparity between the Cubie (2+6) configuration is revealed by the worse non-linear scaling than the Orange Pi (4+4) configuration. In this instance, the

TABLE XIII
ARCHITECTURAL METRICS DERIVED FROM SEM ESTIMATES

Metric	Orange Pi 5	Cubie A7A	Ratio (C/O)
Complexity Sensitivity ($\sum \beta_{\mathtt{M}_c^i}$)	1.659	1.322	0.797
Inter-Island Scaling Friction ($\mathtt{M}_c \times I_L$)	0.350	1.111	3.174
Multi-Core Saturation ($\Delta\beta C_3 \rightarrow C_4$)	+0.023	-0.055	-0.418
Stochastic Coupling Significance (p value)	0.115	0.003	0.026
Critical Complexity Index (χ)	1.417	2.852	2.013

scheduler must prioritize isolation (using only Big cores) over parallelism (using all cores) to avoid bus poisoning.

VII. CONCLUSIONS AND FUTURE WORK

An important goal of modeling efforts in general is to represent the results of empirical evidence faithfully. While asymptotic analysis does this, in the limit where the terms of interest grow arbitrarily large, it is limited in its ability to be informative over term ranges that are seen in practice. This paper illustrates that point for matrix-matrix multiplication, an important computation in many AI/ML applications. All three terms of a cubic model, the linear term, the quadratic term, and the cubic term, are rigorously shown to be significant over a wide range of matrix sizes.

It also would be naïve to treat the terms of interest in a model as if they were independent (a frequent modeling approach). Using the techniques of structural equation modeling, we have shown parameter interactions to be distinctly relevant in predicting execution time.

Finally, based upon the models described above, we have introduced several new performance indicators and applied them to our matrix-matrix multiplication computation executing on a pair of ARM big.LITTLE edge devices. They enable some interesting observations about how the devices are both similar and distinct.

Clearly, this work is but an introduction to these issues. Generalizing to additional applications (beyond matrix-matrix multiplication) and to additional execution platforms (e.g., newer Intel server-class architectures that support power and efficiency cores) is a useful next set of steps. While we contend that our matrix-matrix multiplication target is of interest due in part to its extensive use in AI/ML workloads, those workloads have many other computations incorporated as well.

In addition, while we draw conclusions about our application and execution platforms from the newly presented performance indicators, these indicators warrant additional investigation to assess their robustness as well as their generality.

REFERENCES

[1] R. H. Hoyle, *Structural Equation Modeling: Concepts, Issues, and Applications*. Thousand Oaks, CA: Sage, 1995.

[2] R. B. Kline, *Principles and Practice of Structural Equation Modeling*. New York, NY: Guilford Press, 2010.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[4] H. Wang and D.-Y. Yeung, "A survey on Bayesian deep learning," *ACM Computing Surveys*, vol. 53, no. 5, pp. 108:1–108:37, 2020.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[6] S. Fortune and J. Wyllie, "Parallelism in random access machines," in *Proc. of 10th ACM Symp. on Theory of Computing*, 1978, pp. 114–118.

[7] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[8] J. A. Rico-Gallego, J. C. Díaz-Martín, R. R. Manumachu, and A. L. Lastovetsky, "A survey of communication performance models for high-performance computing," *ACM Computing Surveys*, vol. 51, no. 6, pp. 126:1–126:36, 2019.

[9] H. Xia, Z. Zheng, Y. Li, D. Zhuang, Z. Zhou, X. Qiu, Y. Li, W. Lin, and S. L. Song, "Flash-LLM: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity," *Proc. VLDB Endowment*, vol. 17, no. 2, p. 211–224, Oct. 2023.

[10] Y. Faqir-Rhazoui and C. García, "SYCL in the edge: performance and energy evaluation for heterogeneous acceleration," *The Journal of Supercomputing*, vol. 80, no. 10, pp. 14 203–14 223, 2024.

[11] C. Trott, L. Berger-Vergiat, D. Poliakoff, S. Rajamanickam, D. Lebrun-Grandie, J. Madsen, N. Al Awar, M. Gligoric, G. Shipman, and G. Womeldorff, "The Kokkos EcoSystem: Comprehensive performance portability for high performance computing," *Computing in Science & Engineering*, vol. 23, no. 5, pp. 10–18, 2021.

[12] K. Zhu, Y. Gao, Y. Zhao, L. Zhao, G. Zuo, Y. Gu, D. Xie, Z. Ye, K. Kamahori, C.-Y. Lin *et al.*, "NanoFlow: Towards optimal large language model serving throughput," in *Proc. of 19th USENIX Symposium on Operating Systems Design and Implementation*, 2025, pp. 749–765.

[13] S. D. Harris, R. D. Chamberlain, and C. D. Gill, "Performance models for non-uniform heterogeneous platforms and their validation using structural equation modeling," *International Journal of Networking and Computing*, vol. 16, no. 1, pp. 20–51, Jan. 2026.

[14] P. Stoica and Y. Selen, "Model-order selection: a review of information criterion rules," *IEEE Signal Processing Magazine*, vol. 21, no. 4, pp. 36–47, 2004.