A Taxonomy of Application Properties for Mixed-Precision Autotuning

Gülçin Gedik and Robert Schöne
ZIH, CIDS, TU Dresden, Germany
gulcin.gedik@mailbox.tu-dresden.de, robert.schoene@tu-dresden.de

I. Motivation

Modern HPC platforms deliver peak throughput and energy efficiency in reduced-precision formats (FP16 and below), driven largely by machine-learning workloads. Many scientific applications, however, still default to FP64 even where portions of their computation could tolerate lower precision without compromising scientifically relevant results [1, 3]. Mixed-precision computing—using multiple floating-point formats within an application—promises faster computation and lower energy consumption by exploiting these hardware capabilities. Tool-based mixed-precision autotuners aim to automate the otherwise tedious process of identifying which variables, loops, or kernels can safely operate at reduced precision.

Yet autotuning succeeds unevenly across real applications: weather and climate codes often yield substantial gains, while molecular dynamics or computational mechanics codes prove fragile or inconclusive [2, 5, 6, 7]. A fundamental question remains underexplored: *which properties of a scientific application make it a good or poor candidate for tool-based mixed-precision tuning?* Existing discussions focus predominantly on tool features rather than on why specific applications resist tuning regardless of the tool used.

II. A Three-Layer Analytical Framework

We ground our analysis in a three-layer model of in-place mixed-precision autotuning (Fig. 1). The bottom layer identifies *design levers* (granularity, validation oracle, integration stage, objectives) that define the configuration space. The middle layer captures *primary cost drivers* (instrumentation overhead, validation runs, cast/compiler interference, portability risk) that determine end-to-end tuning effort. The top layer shows the *workflow phases* where autotuning proceeds: profiling, sensitivity analysis, search-space reduction, search and evaluation, and transformation. Design levers directly shape which cost drivers become dominant, and application properties constrain both.

III. Taxonomy of Application Properties

We propose a taxonomy that organizes application properties into four categories corresponding to the primary dimensions along which mixed-precision tools must reason [4].

Structural properties describe how computation is organized—data layout, control flow, and parallelization strategy. They determine tuning granularity (per kernel, loop, variable, or instruction), analysis cost, and cast density. Codes with cleanly separated kernels and regular access patterns enable coarse-grained decisions that reduce search cost, while monolithic or irregular structures inflate it.

Behavioral properties capture how an application executes and how its outputs vary across runs. Fixed-iteration programs allow final-output validation, whereas convergence-based termination requires tracking trajectories for stalls, false convergence, or instability. Non-deterministic applications demand statistical validation over multiple runs, sharply increasing cost.

Numerical and domain properties govern error sensitivity and correctness criteria. Well-conditioned kernels with built-in recovery mechanisms (e.g., iterative refinement) permit aggressive exploration, while ill-conditioned operations or strict domain invariants force conservative pruning. Domain constraints—statistical fidelity in climate models versus traceable reproducibility in safety-critical codes—further restrict the feasible search space.

Performance and architectural properties determine whether reduced precision actually delivers gains on a given platform. Compute-bound regions can exploit low-precision units; memory-bound codes benefit from reduced data movement. However, conversion overhead (casting between formats) can absorb or exceed the theoretical gains, as demonstrated by the false-friend examples in our poster where a stencil kernel's bottleneck shift from memory- to compute-bound under FP16 causes net slowdown.

IV. Impact Matrix

Table I summarizes how property categories constrain each workflow phase. This *impact matrix* makes explicit why the same tuning stage can be trivial for one application and prohibitive for another, and why tool comparisons without an explicit application-property frame can be misleading.

V. Decision Gates and Stress Case

We operationalize the taxonomy as five qualitative *decision gates* for assessing application readiness: (i) tuning atoms exist, (ii) a correctness oracle is available, (iii) behavioral variability is bounded, (iv) recovery mechanisms exist, and (v) the dominant bottleneck supports payoff. An application is well-suited for current autotuning tools when most gates pass.

As a stress case, consider an unstructured finite-element Krylov solver with sparse matrix-vector product. It passes only

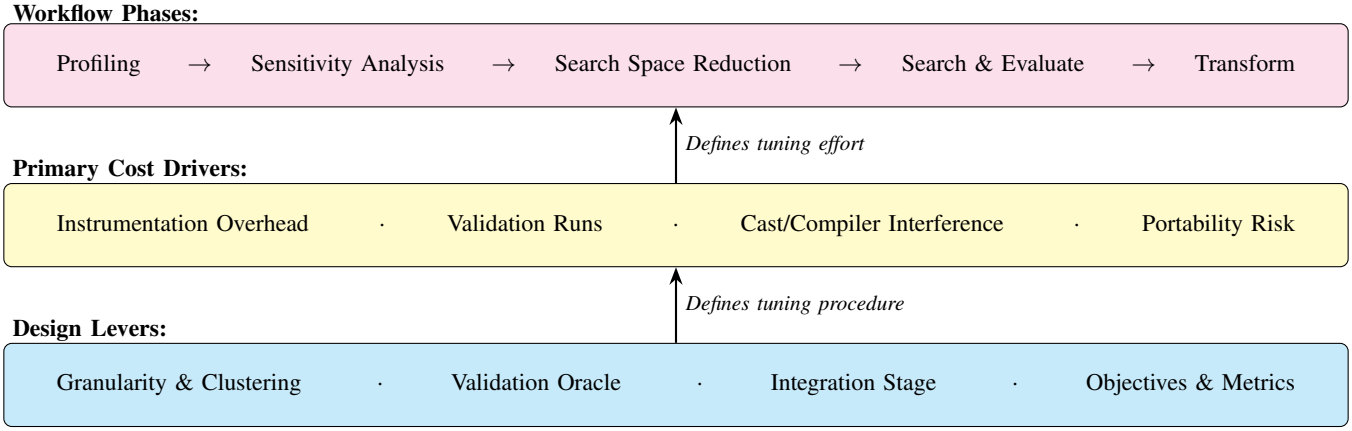


Fig. 1. Three-layer model of mixed-precision autotuning: design levers shape cost drivers, which manifest across workflow phases. Application properties constrain all three layers.

TABLE I
IMPACT MATRIX: HOW PROPERTY CATEGORIES CONSTRAIN WORKFLOW PHASES (ABBREVIATED).

	Profiling	**Sensitivity**	**Space Reduction**	**Search & Evaluation**
Structure	Hotspot atoms	Static vs. dynamic analysis	Granularity, clustering	Cast density, search cost
Behavior	Input regimes	Outputs vs. trajectories	Conservative if input-dep.	Single-run vs. statistical
Numerical	Signif. variables	Recovery vs. amplification	Pruning aggressiveness	Domain-aware objectives
Performance	Bottleneck class	Instrumentation overhead	HW-aware format pruning	Time/energy metrics

the oracle gate (residual-based verification is standard), but fails the remaining four: deep, data-dependent call structures prevent clean tuning-atom extraction; convergence-based termination and run-to-run variance require expensive statistical validation; error accumulating across iterations limits safe pruning; and memory-bound execution makes cast overhead salient. This 4/5 gate failure composition predicts expensive profiling and validation, limited safe pruning, and fragile payoffs—illustrating why suitability must be judged holistically rather than by any single property.

VI. Significance for the SC Community

This work shifts the mixed-precision discourse from tool-centric feature lists to application-centric explanations of when and why tools succeed or fail. For the SC community, this perspective offers three practical contributions: (1) a vocabulary for stating tool assumptions and limitations against concrete application properties, (2) a principled basis for benchmark selection—suites that span the property space enable fairer tool comparisons, and (3) actionable criteria that help application developers set realistic expectations before investing in precision tuning. By making the interplay between application characteristics and autotuner feasibility explicit, we aim to reduce wasted tuning effort on unsuitable codes and direct research toward the most impactful challenges.

Acknowledgments

The authors gratefully acknowledge the funding and support of the NHR Graduate School of the NHR-Verein e.V., provided through the German BMFTR and the Free State of Saxony under Grant No. NHR2025SN.

References

[1] Qinglei Cao, Sameh Abdulah, Hatem Ltaief, Marc G. Genton, David Keyes, and George Bosilca. Reducing data motion and energy consumption of geospatial modeling applications using automated precision conversion. In *2023 IEEE International Conference on Cluster Computing (CLUSTER)*, 2023. DOI: 10.1109/CLUSTER52292.2023.00035.

[2] Yanxiang Chen, Pablo de Oliveira Castro, Paolo Bientinesi, Niclas Jansson, and Roman Iakymchuk. Enabling mixed-precision in spectral element codes. *Future Generation Computer Systems*, 2026. DOI: 10.1016/j.future.2025.107990.

[3] Jack Dongarra, John Gunnels, Harun Bayraktar, Azzam Haidar, and Dan Ernst. Hardware trends impacting floating-point computations in scientific applications, 2024. DOI: 10.48550/arXiv.2411.12090.

[4] Gülçin Gedik, Christian von Elm, and Robert Schöne. A taxonomy of application properties for mixed-precision autotuning. In *Proceedings of the 2026 ACM/SPEC International Conference on Performance Engineering*, 2026. DOI: 10.1145/3777911.3801102.

[5] Pradeep V Kotipalli, Ranvijay Singh, Paul Wood, Ignacio Laguna, and Saurabh Bagchi. Ampt-ga: automatic mixed precision floating point tuning for gpu applications. In *Proceedings of the ACM International Conference on Supercomputing*, 2019. DOI: 10.1145/3330345.3330360.

[6] Bálint Siklósi, Pushpender K. Sharma, David J. Lusher, István Z. Reguly, and Neil D. Sandham. Reduced and mixed precision turbulent flow simulations using explicit finite difference schemes. *Future Generation Computer Systems*, 2026. DOI: 10.1016/j.future.2025.108111.

[7] Jackson Vanover, Alper Altuntas, and Cindy Rubio-González. Toward automated precision tuning of weather and climate models: A case study. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2024. DOI: 10.1109/SCW63240.2024.00026.