Technical Brief

CPU Performance Analysis
Workload Benchmarking

# Active Benchmarking for Better Performance Predictions

**Workload similarity analysis can help identify more meaningful benchmarks for IT purchasing decisions.**

*By Harshad Sane, Principal Software Engineer, Intel*

## Up your benchmarking game

Intel® processors have built-in mechanisms for monitoring performance at a granular level. You can collect this performance data and use it to identify which benchmarks are most similar to your real-world workloads in how they exercise different system components. Benchmarks with greater similarity to your workloads provide a better way to identify systems that will perform optimally running those workloads.

## One benchmark does not fit all workloads

The high-tech industry has traditionally relied on one or two benchmarks for a simple, standardized way to assess CPU performance and, by extension, overall system performance. IT decision makers (ITDMs) use these benchmarks to choose the systems that best meet their price and performance requirements. But performance requirements become more complex as workloads grow increasingly diverse—and often more distributed, which brings memory and network complexities of its own.

A system that performs well for traditional, general-purpose computing might not be optimal for modern, cloud-native applications and other data-intensive workloads that rely on system components beyond the CPU. A microservices architecture can exacerbate the benchmarking challenge because of the number of moving parts and the cascading latencies among those parts.

One step toward more effective benchmarking is to move beyond CPU benchmarks like SPECrate 2017 Integer (SPECint)[1] to also consider proxy benchmarks that target specific workloads. These could include LINPACK for supercomputing and the collection of TPC benchmarks for databases and transaction processing systems. DeathStarBench is another example of a workload-specific benchmark for microservices that exercises much more than just core compute. But even these benchmarks might not accurately reflect a system's performance characteristics in a production setting.

Intel proposes a more systematic approach, which includes:

- Gathering telemetry data to better understand performance needs and hot spots for a real-world workload
- Selecting an appropriate benchmark that will exercise the correct mix of system components

This approach advocates choosing a benchmark with a similar "fingerprint" to your real-world workload to help you more accurately identify systems on which your workload will perform well.

## Fundamentals of performance analysis

Sophisticated methods have been developed for CPU performance analysis using telemetry data from performance monitoring units (PMUs), which capture underlying microarchitectural events using hardware counters. These events can be used to understand system behavior and the impact of an application's footprint on various aspects of the CPU, such as cache, memory, and translation lookaside buffers (TLBs).

In the past, the primary purpose of such performance analysis has been for application developers to pinpoint critical bottlenecks in application execution to optimize their applications for particular CPUs. However, Intel proposes a different use for the same kind of analytics. This use can help IT organizations more accurately predict the capabilities of different CPUs and systems to run existing real-world applications and workloads.

Every workload tends to use the components of the CPU in a way that can be captured by PMU telemetry and characterized as the workload's fingerprint. You can use performance analytics to compare the fingerprint of a real-world workload to the fingerprints of different benchmarking systems in order to identify the best benchmarking match for the workload.

Tools such as PerfSpect enable the collection of PMU telemetry from the CPU while running a workload. One difficulty arises from the sheer quantity of data generated by the PMUs and collected by the tools. It can be difficult to find the most meaningful information amidst the mountain of performance data available. To solve this problem, Intel turns to a top-down approach to performance analysis to help cut through the noise of excess data.

The idea is that to pinpoint a performance bottleneck, developers will first identify, at a high level, where the processor is spending its time within the complex execution pipeline. They can then drill down through a hierarchical classification tree to pinpoint the precise location of the bottleneck, as shown in Figure 1.

For the purpose of characterizing workloads, however, there is no real need to pinpoint specific bottleneck locations. *Patterns* of performance are what define a workload fingerprint, and it turns out the top-level classifications—front-end bound and back-end bound—provide sufficient information to identify such patterns.

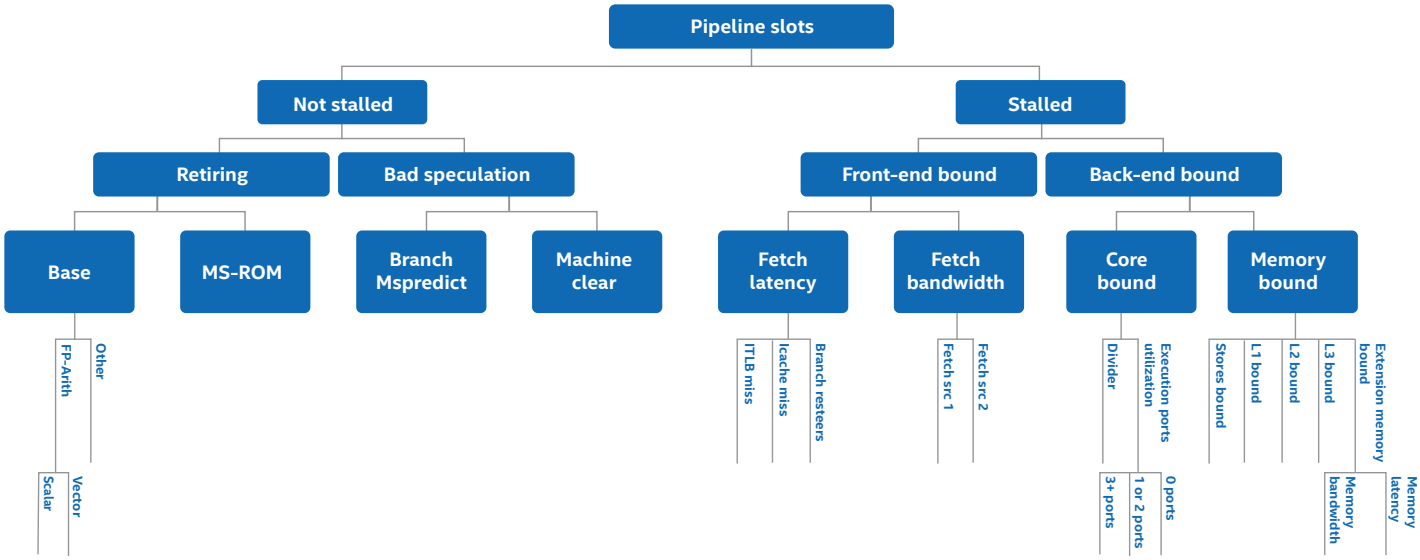


**Figure 1**. A hierarchical classification tree illustrates top-down performance analysis

## Intel PMU: A rich source of performance counters

The Intel PMU is hardware built inside a processor to measure its performance parameters, including instruction cycles, cache hits, cache misses, branch misses, and many others. The Intel PMU provides so many performance counters, in fact, that it can be challenging to sift through all the data to find the most useful information. The data generated by the Intel PMU is incredibly rich, and the method of compressing it and extracting insights described in this paper is a powerful way to harness the large amount of available data meaningfully. Learn more about the Intel PMU and performance counters at perfmon-events.intel.com.

## Workload fingerprinting

There are two parts to the methodology for workload fingerprinting. First, the dataset is analyzed to determine which two variables, or components, are the principal components that best represent the whole of the data in characterizing different workloads. Second, workloads are plotted on the axes of the two principal components so that the similarity of workloads can be seen by their proximity on the chart.

## Principal component analysis

An important concept for workload fingerprinting is principal component analysis (PCA), a method to reduce the number of variables of a dataset while preserving as much information as possible. PCA allows you to visualize large datasets through a reduced set of features in a compressed format. You might think of it like compressing a large bitmap of a photograph into a much smaller JPEG file—some detail is lost if you zoom in, but the big picture remains clear.

The idea is to identify the top components that account for the greatest differentiation among workloads. Intel found that the two principal components that best differentiate workloads are the degree to which a workload is front-end bound on one hand, and the degree to which it is back-end bound on the other.

- **Front-end bound**: Stalls occur in fetching instructions from memory and translating them into micro-operations (µops).
- **Back-end bound**: Stalls occur in scheduling and executing (that is, "retiring") those µops.

Ideally, the front end is always busy fetching and translating instructions and the back end is always busy executing µops. Inefficiency occurs when the two are not in balance. When a system is front-end bound, instructions cannot be fetched and queued as µops fast enough to keep the back end busy. When a system is back-end bound, the successful execution of µops is not fast enough to keep the front end busy.

The fact that a system can be both front-end bound and back-end bound, each to a greater or lesser degree, makes it possible to plot a workload on those two axes as a way of fingerprinting the workload on the system.

## Workload similarity analysis

By plotting workload and benchmark characteristics on the two axes of the principal components, you can determine mathematically (and visually) how similar the components' fingerprints are to each other. Figure 2 shows conceptually how workloads with similar fingerprints cluster together on the chart.

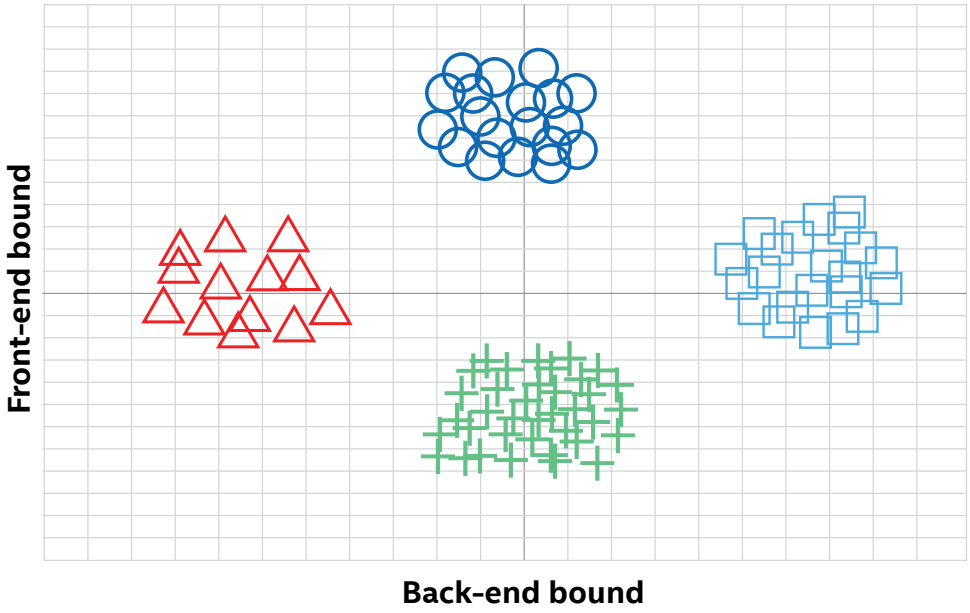


**Figure 2**. Conceptual mockup showing clusters of similar workloads plotted on two principal component axes

Intel used PerfSpect to extract PMU information and process top-down metrics for dozens of real-world customer workloads and benchmarking tests. It then used these metrics as input data to run through PCA, which consists of several mathematical stages of normalization, eigenvector generation, and so on. This produced the representative principal components for each workload, plotted in Figure 3.

Figure 3 positions workloads that are more back-end bound farther to the left, and those that are more front-end bound closer to the bottom. Workloads near each other on this chart show similar microarchitectural behavior, while larger distances indicate a higher dissimilarity of workloads. Figure 3 draws lines around groups of workloads as a visual aid to highlight the clustering effects. Clearly the real-world customer workloads (red) were much more back-end bound than the SPECint benchmarks (blue). The customer workloads aligned much more closely with the cloud-native benchmarks (green), though customer workloads tend to be somewhat less front-end bound.

Intel completed similarity analysis for dozens of workloads and benchmarks in order to demonstrate the usefulness of the method. A systems analyst would not need to plot so many data points, but might be interested in comparing the similarities of one or two heavy workloads to a few benchmarks already in use or under consideration.

The fact that SPECint forms a cluster very different from the real workloads shows the importance of identifying proxy workloads that exhibit similar properties.

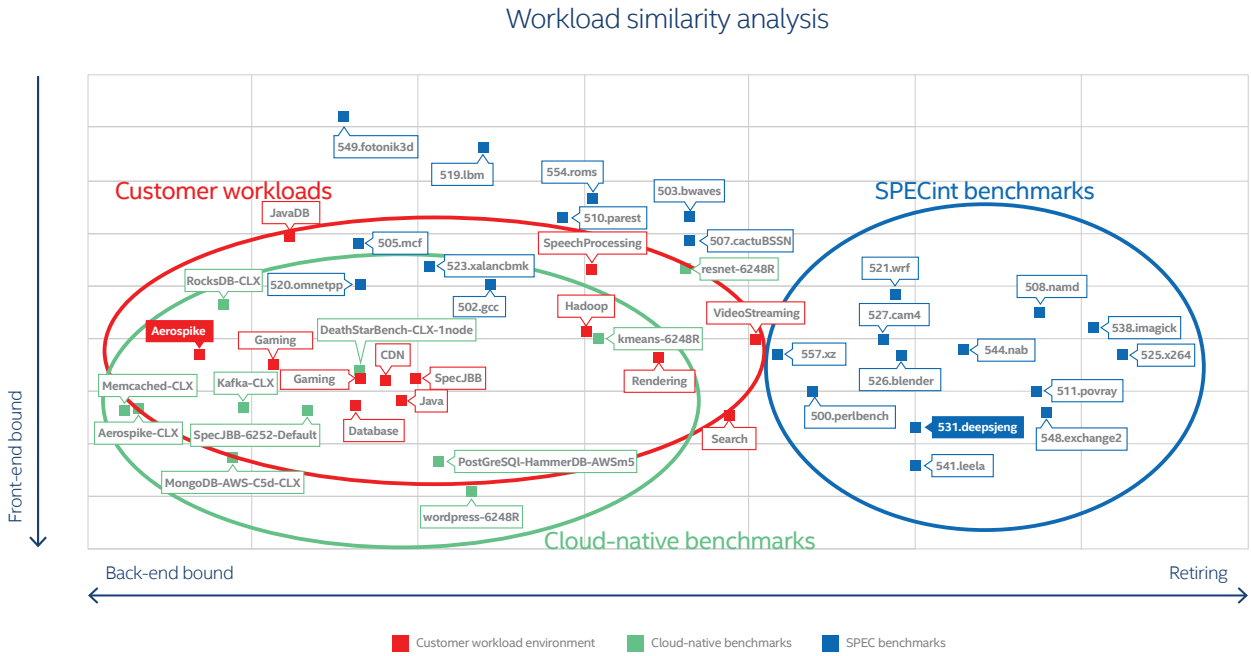


**Figure 3**. Real-world workloads and benchmarks cluster by similarity

## Example case

Consider one pair of data points from the set in Figure 3 as a representative example of the kind of similarity analysis a system architect might want to perform. Suppose the customer using Aerospike, an in-memory open source NoSQL database-management system (DBMS), wants to purchase a new SKU for the data center to run that workload. Perhaps SPECint is the benchmark system that the system architect used in the past for such decisions, but she wants to determine if it's the most appropriate benchmark for this workload. Upon running a similarity analysis on the Aerospike workload and the 531.deepsjeng component of SPECint, a chess application tree-search benchmark, it becomes apparent that the Aerospike workload and 531.deepsjeng are on opposite ends of the back-end bound spectrum. The Aerospike workload is highly back-end bound compared to the benchmark. This would be a good clue for the system architect to look for a more fitting benchmark.

But this system architect might want to better understand the nature of the mismatch between the Aerospike workload and the benchmark test. And here it's important to note that while the chart plots similarity on only two dimensions, those axes are representative of a large number of data points. All the PMU telemetry data was collected, so it is possible to drill down into the details.

**Table 1**. Metrics behind the similarity analysis show where the differences are greatest

| Metric | 531.deepsjeng | Aerospike |
|---|---|---|
| cpu_operating_frequency_GHz | 2.92 | 2.70 |
| cpu_utilization_% | 96.65 | 100.01 |
| cpi | 0.91 | 3.88 |
| frontend_bound_% | 24.93 | 24.97 |
| backend_bound_% | 5.79 | 54.50 |
| L1_bound_% | 16.65 | 17.36 |
| L2_bound_% | 1.00 | 2.21 |
| L3_bound_% | 0.95 | 5.29 |
| mem_bound_% | 4.37 | 11.39 |
| mem_bandwidth_% | 0.00 | 0.27 |
| mem_latency_% | 4.54 | 11.49 |
| retiring_% | 53.57 | 18.30 |

Table 1 shows some of the metrics used to generate the principal components. The table shows that although both workloads run at the same frequency and CPU utilization percentage and are similarly front-end bound, the Aerospike workload is highly back-end bound by comparison. Moreover, most of the back-end inefficiency can be traced down to being memory-bound, and especially memory-latency-bound compared to the SPECint component.

The takeaway is that although the CPU is exercised at the same level, the components within the CPU are stressed in completely different ways. This helps the systems architect understand at a more granular level why that particular benchmark test is not ideal for evaluating how well different systems will perform on the company's Aerospike workload.

## Conclusion

IT organizations using benchmark tests to evaluate hardware system purchases would be well advised to assess their benchmarks first. Today's workloads are complex, and a broad spectrum of components will contribute to the optimal performance for any given workload. If a benchmark is not similar to a real-world workload, the benchmark results will mean little. If you choose a benchmark with a similar footprint to your target workload, and then use that similar benchmark to evaluate alternative CPUs and systems, then high performance on that benchmark will be a good predictor of optimized performance on your real-world workload.

The method demonstrated in this paper for conducting workload similarity analysis is not just theoretical. The code has been developed, and you can run your own workload-similarity analyses using a collection of open source tools at https://github.com/intel/PerfSpect.

## Learn more

**Read the white paper, "A Top-Down Method for Performance Analysis and Counters Architecture," at https://www.researchgate.net/publication/269302126_A_Top-Down_method_for_performance_analysis_and_counters_architecture, written by Ahmad Yasin, Principal Engineer at Intel and originator of the TMA method featured as a part of this analysis.**

**Find out more about how Intel provides greater flexibility in how you utilize the cloud, optimize costs, and improve efficiency:**

- **Cloud tools: intel.com/content/www/us/en/cloud-computing/cloud-tools.html**
- **Cloud computing overview: intel.com/content/www/us/en/cloud-computing/overview.html**
- **Similarity Analyzer: https://github.com/intel/PerfSpect/tree/master/similarity-analyzer**

[1] SPEC is a registered trademark of Standard Performance Evaluation Corporation. See https://spec.org/spec/trademarks.html.



Printed in USA 0322/DR/PRW/PDF Please Recycle 350211-001US