International Journal of Multidisciplinary Research and Modern Education (IJMRME)
Impact Factor: 7.315, ISSN (Online): 2454 - 6119
(www.rdmodernresearch.org) Volume 8, Issue 2, 2022

HARDWARE CONFIGURATION AND EXECUTION TIME VARIABILITY IN SCIENTIFIC COMPUTING RESEARCH BENCHMARKS ACROSS CLOUDLAB SERVERS

Divya Vishwanath* & Paul Johnson Asamoah**
* Janardhan Rai Nagar Rajasthan Vidyapeeth, Udaipur, Rajasthan, India
** Power Life University College, Accra, Ghana, West Africa

Cite This Article: Divya Vishwanath & Paul Johnson Asamoah, "Hardware Configuration and Execution Time Variability in Scientific Computing Research Benchmarks Across CloudLab Servers", International Journal of Multidisciplinary Research and Modern Education, Volume 8, Issue 2, Page Number 111-120, 2022.

Copy Right: © IJMRME, 2022 (All Rights Reserved). This is an Open Access Article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium provided the original work is properly cited.

Abstract:

This analysis examines how execution settings relate to performance variability in scientific computing benchmarks, addressing the reliability of computational comparisons across servers. It uses 120,564 CloudLab executions recorded between 2018 and 2019, aggregated into 6,588 server and configuration groups across 183 servers. Matched comparisons estimate differences in execution time coefficients of variation within the same server and benchmark, with confidence intervals calculated across server averages. Enabled frequency scaling is associated with 0.130 percentage points higher variability, while the 20-thread setting is associated with 0.137 percentage points higher variability than one thread. Both confidence intervals exclude zero under the stated inference assumptions. Benchmark comparisons reveal different patterns, including a negative average thread difference for CG, indicating that the overall association is not uniform across workloads. Sensitivity checks retaining groups with at least 10 executions and excluding the server with the highest mean variability preserve the positive average differences. The findings support workload specific performance evaluation and encourage computing operators to report dispersion alongside mean execution time. The associations do not establish causal mechanisms or numerical reproducibility.

Key Words: Benchmarking, Execution Time Variability, Frequency Scaling, Scientific Computing, Thread Count.

1. Introduction:

The credibility of scientific computing depends on whether performance measurements remain informative across repeated executions and computing environments. Execution time can vary even when the same benchmark is run on machines with comparable configurations. Such variation complicates comparisons between systems and can make a measured performance difference difficult to distinguish from ordinary execution variability. Chen et al., 2015, show that computer comparisons can be misleading when they disregard the distribution of performance measurements. For researchers and computing operators, the substantive question is therefore not only how quickly a workload runs, but also how consistently it runs under identifiable conditions.

Performance variability arises from interactions between a workload and its execution environment. A benchmark uses processing resources according to its computational requirements, while server conditions and execution settings shape how those resources are made available. Zhang et al., 2017, examine variability in parallel workloads and demonstrate the importance of understanding how execution conditions relate to performance. Pusukuri et al., 2012, further show that scheduling and memory allocation policies can influence variation across repeated executions. These studies establish plausible sources of variability, but the presence and strength of a particular mechanism cannot be inferred without measuring the relevant operating conditions.

Three observable characteristics are relevant to this problem. Server identity distinguishes the machines on which benchmarks run. Frequency scaling identifies whether dynamic voltage and frequency scaling is enabled, while thread count identifies the number of threads used for execution. Each characteristic may be associated with execution time variability, but their implications need not be uniform across workloads. Rauber et al., 2018, show that applications respond differently to processor frequency and thread configurations. Morgan et al., 2021, also report variation across computing nodes in parallel scientific calculations. Together, these findings support comparisons that retain benchmark identity and distinguish server differences from configuration differences.

The outcome also requires a measurement that distinguishes variability from execution speed. This analysis uses the coefficient of variation of execution time, defined as the sample standard deviation divided by the mean and expressed as a percentage. The measure describes dispersion relative to the average duration of repeated executions. It enables comparisons of relative variability across benchmark groups, although its precision depends partly on the number of executions and the periods covered. Irving et al., 2020, emphasize the need to examine performance distributions and uncertainty when comparing computer systems. The coefficient

of variation addresses relative dispersion; it does not determine whether numerical outputs agree across executions.

Existing research has documented performance variation across workloads, machines and cloud environments. Laaber et al., 2019, report substantial differences in variability across benchmarks and computing environments, while Duplyakin et al., 2020, examine changes over time in CloudLab performance. These contributions establish that variation deserves attention, but the reviewed evidence does not by itself determine how frequency scaling and thread count compare within the specific server and benchmark groups examined here. The analytical contribution is a matched comparison of these observed settings, combined with a descriptive assessment of differences across server identities. This design qualifies general performance claims by examining whether an average configuration difference is consistent across the recorded benchmarks.

This study aims to examine how server identity, frequency scaling and thread count relate to execution time variability in scientific computing benchmarks. The central research question is: How does execution time variability differ across servers and execution settings under comparable benchmark conditions? The analysis addresses this question through three objectives: to describe variability across server identities, to compare enabled and disabled frequency scaling within matched configurations, and to compare one thread with 20 threads within matched configurations.

The empirical setting comprises 120,564 recorded executions from the CloudLab CPU benchmarking dataset, aggregated into 6,588 server and configuration groups across 183 servers. The observations extend from 15 August 2018 to 31 October 2019. For each server, the analytical sample contains nine benchmarks under two frequency scaling settings and two thread counts. Matched comparisons hold server identity, benchmark identity and the remaining recorded setting constant, while uncertainty is assessed across server averages. The approach supports associations within the observed sample; it does not identify the physical causes of variability or establish causal effects.

The results indicate higher average execution time variability for enabled frequency scaling and for 20 threads, while the direction and magnitude of differences vary across benchmarks. These findings inform a specific performance evaluation decision: comparisons of execution settings should retain workload identity and report variability alongside average execution time. Their relevance extends to computing environments where repeated benchmark measurements guide configuration choices, but the magnitude of the observed associations remains specific to the recorded CloudLab servers and workloads.

2. Theory, Conceptual Framework and Research Questions:

Execution time in scientific computing emerges from interactions among the benchmark workload, the server and the execution settings. Performance evaluation therefore requires attention to both the average time taken and the variation observed across repeated runs. Papadopoulos et al., 2021, identify controlled measurement and transparent reporting as foundations of reproducible performance evaluation. Laaber et al., 2019, show that performance variability can differ substantially across benchmarks and computing environments. These findings support an analysis that compares execution time variability within comparable benchmark conditions.

This analysis uses the coefficient of variation as the outcome. It is the sample standard deviation of execution time divided by the mean execution time, expressed as a percentage. The measure captures the dispersion of repeated execution times relative to their mean. Each observation represents one combination of server, benchmark, frequency scaling setting and thread count. Benchmark identity is included in comparisons because workloads differ in their use of computing resources.

Server Identity:

Server identity distinguishes the individual machines on which benchmarks were executed. Even when servers belong to the same class, their repeated execution times may differ because each machine operates under conditions that can vary across runs. The server identifier provides a means of examining such differences without treating the identifier as a measured hardware specification.

Morgan et al., 2021, found variation in the performance of parallel scientific computations across computing nodes. However, differences between nodes do not establish that a particular hardware characteristic caused the observed variation. In this analysis, comparisons are made across server identities while accounting for the benchmark, frequency scaling setting and thread count.

Research question 1: How does execution time variability differ across server identities under comparable benchmark and execution settings?

Dynamic Voltage and Frequency Scaling:

Dynamic voltage and frequency scaling changes processor operating conditions in response to system requirements. The available data distinguish runs with the setting enabled from runs with it disabled. A change in processor frequency can alter execution time, but its relationship with relative variability depends on how a workload responds to frequency changes and other sources of variation.

Rauber et al., 2018, demonstrate that frequency and thread configuration have different performance implications across applications. Their findings establish the importance of workload characteristics but do not

establish a common direction for the relationship between frequency scaling and the coefficient of variation. The analysis therefore compares the two observed settings without prespecifying that either setting produces greater variability.

Research question 2: How does execution time variability differ between enabled and disabled frequency scaling settings for comparable servers, benchmarks and thread counts?

Thread Count:

Thread count represents the number of threads used to execute a benchmark. The available observations distinguish runs using one thread from those using 20 threads. Additional threads can distribute work across processing resources, while coordination and synchronization can expose execution time to delays affecting individual threads.

Akkan et al., 2013, explain how operating system interruptions can delay parallel tasks at synchronization points. Rauber et al., 2018, also show that the performance response to additional threads depends on the application. These mechanisms do not establish that the 20-thread setting must produce a higher coefficient of variation for every benchmark. The analysis therefore examines the observed difference under comparable server, benchmark and frequency scaling conditions.

Research question 3: How does execution time variability differ between one thread and 20 threads for comparable servers, benchmarks and frequency scaling settings?

3. Data:

This section establishes the provenance of the benchmark data, defines the analytical sample and documents the construction of execution time variability. It also specifies the observed variables and their alignment with the research questions.

3.1 Data Sources, Population, and Sample Construction:

The analysis uses the CPU benchmark file cpu_xl170.csv from the CloudLab benchmarking dataset provided by Duplyakin, 2020. The file is available through Zenodo as version 6 under the identifier 10.5281/zenodo.3686952. The repository reports that the benchmark records were exported from the source database in February 2020. The file is openly accessible, although the applicable licence is not specified in the available repository record.

The target population comprises repeated scientific benchmark executions on CloudLab servers. The accessible sampling frame consists of the CPU observations in the specified file. The recorded execution period extends from 15 August 2018 to 31 October 2019, using Coordinated Universal Time. Observations occur at irregular execution times rather than at a fixed daily or annual frequency.

The analytical unit is a server, benchmark, frequency scaling setting and thread count combination. The dataset contains all 36 observed combinations for each of 183 servers: nine benchmarks, two frequency scaling settings and two thread counts. This configuration structure is balanced, although the number of recorded executions varies across groups. The data support comparisons within benchmark conditions, consistent with the emphasis on comparable measurement conditions in Papadopoulos et al., 2021, and Laaber et al., 2019.

Sample Construction Stage	Servers	Observations	Rule
Accessible CPU file	183	120,564 executions	Recorded observations in cpu_xl170.csv
Missing values in source fields used	0 affected servers	0 executions	No missing values observed in the extracted source fields
Aggregated analytical sample	183	6,588 groups	Grouped by server, benchmark, frequency scaling setting and thread count
Sample available for statistical estimation	183	6,588 groups	No model has yet been estimated

Each group contains between 3 and 25 executions. The recorded sample is a subset of CloudLab benchmark activity; it is not a census of scientific computing systems.

3.2 Data Integration, Cleaning, and Quality Assurance:

The analytical file was constructed from one source file. No external dataset was merged. Server identifier, benchmark identifier, frequency scaling setting and thread count formed the grouping keys. Repeated executions within a group were retained to calculate the distribution of execution times. The resulting file contains no duplicate grouping keys.

The extracted source fields contain no missing values, and all 6,588 analytical groups have values for the outcome and the observed explanatory variables. No imputation, interpolation or outlier removal was applied. Execution times were retained as recorded, and group statistics were rounded to six decimal places after calculation.

The file retains the earliest and latest Unix timestamps for each group. These timestamps define the observed span of each group, but the aggregated file does not retain the timing of every individual execution. The number and timing of repetitions differ across groups. These differences matter because variability

estimates based on fewer executions can be less precise. Morgan et al., 2021, show that performance variation can differ across computing nodes and execution conditions, supporting attention to the distribution of repeated measurements.

3.3 Variable Measurement, Construction, and Validation:

Execution time variability is the dependent variable. It is measured by the coefficient of variation of execution time within each analytical group. This measure expresses the sample standard deviation as a percentage of the group mean:

$$CV_i = 100 \times \frac{S_i}{\overline{T}_i} \qquad 1$$

In Equation 1, CV_i is the execution time coefficient of variation for group i, S_i is the sample standard deviation of execution time, and $\overline{T}_i$is the mean execution time. The sample standard deviation uses one fewer than the number of executions in its denominator. A higher coefficient indicates greater execution time dispersion relative to the mean.

Variable	Analytical Role	Definition and Construction	Source Identifier	Unit and Coverage	Missing Values and Limitation
Execution time variability	Outcome	Sample standard deviation divided by mean execution time, multiplied by 100	Time CV percent	Percentage; 6,588 groups	None observed; precision depends partly on the number of runs
Server identity	Explanatory variable	Identifier of the server executing the benchmark	nodeid	183 servers	None observed; identifies a machine rather than a measured hardware specification
Frequency scaling setting	Explanatory variable	Recorded enabled or disabled setting	dvfs	Two categories; 6,588 groups	None observed; recorded setting does not provide a frequency trace
Thread count	Explanatory variable	Number of threads used for the benchmark	total_threads	One or 20 threads; 6,588 groups	None observed; coverage does not include intermediate thread counts
Benchmark identity	Comparison variable	Identifier of the computational benchmark	testname	Nine categories; 6,588 groups	None observed; workload properties are not separately measured
Run count	Measurement descriptor	Number of execution records in each group	Derived from exec_time records	3 to 25 executions per group	None observed; unequal counts affect the precision of group variability estimates
Mean execution time	Component of outcome	Arithmetic mean of recorded execution times within a group	exec_time	Seconds; 6,588 groups	None observed; all group means are positive
Execution time standard deviation	Component of outcome	Sample standard deviation of recorded execution times within a group	exec_time	Seconds; 6,588 groups	None observed; calculated from repeated executions
First and last timestamps	Coverage descriptors	Earliest and latest execution timestamps within a group	timestamp	Unix seconds; 6,588 groups	None observed; individual execution dates are not retained in the aggregated file

Frequency scaling and thread count describe execution settings. Rauber et al., 2018, show that application performance can respond differently to frequency and thread configurations. Server identity supports comparisons across machines, while benchmark identity preserves the distinction between workloads. The interpretation of thread count also reflects the potential for operating system interruptions and synchronization delays described by Akkan et al., 2013.

Across the 6,588 groups, the mean coefficient of variation is 0.520 percent, the standard deviation is 0.605 percentage points, and the median is 0.373 percent. Values range from 0.019 to 8.096 percent. These statistics describe the distribution of group measures; they do not establish differences between configurations.

3.4 Analytical Framework, Identification, and Reproducibility:

The analytical purpose is associational. The research questions concern differences in execution time variability across server identities, frequency scaling settings and thread counts under comparable benchmark conditions. Server identity is categorical, and the two settings are observed as configuration categories. Benchmark identity is required for comparison because the outcome is calculated separately for each workload.

Comparisons of frequency scaling settings and thread counts can use groups from the same server and benchmark. Observations from the same server are related, so an estimation procedure must account for dependence within servers. Comparisons across server identities require attention to the repeated configuration groups observed for each machine. The number of executions contributing to each coefficient also varies and should inform the interpretation of precision.

No regression model, inference procedure or robustness check has been implemented in the available workbook. The data therefore establish an analytical sample and measurable variables, but they do not yet establish the direction, magnitude or statistical significance of any association. Server identity does not isolate a specific physical cause, and configuration differences alone do not establish causal effects.

The workbook records the grouping variables, run counts, calculated means, sample standard deviations, coefficients of variation and group boundary timestamps. These fields permit review of the analytical definitions alongside the source file. The public source file provides the individual execution records needed to reconstruct the group measures.

4. Methods:

4.1 Research Design and Identification:

The analysis uses a descriptive and associational design to examine execution time variability across server identities and execution configurations. The analytical unit is a server, benchmark, frequency scaling setting and thread count combination. The outcome is the coefficient of variation defined in Equation 1. The sample construction table and variable dictionary identify the source records, grouping rules and measurement units.

Server differences are assessed descriptively by averaging the outcome across the same 36 benchmark and configuration combinations for each server. The resulting summaries describe differences among the observed machines. A server identifier does not measure a specific hardware characteristic, and the analysis does not attribute differences to a physical cause.

The configuration comparisons use matched groups. The frequency scaling comparison pairs enabled and disabled settings for the same server, benchmark and thread count. The thread comparison pairs 20 threads and one thread for the same server, benchmark and frequency scaling setting. Matching holds these recorded categories constant and makes the comparisons specific to a common server and workload.

The estimands are average differences in execution time variability, expressed in percentage points. They are conditional associations rather than causal effects. The available records do not establish randomized assignment or measure all operating conditions that could affect variability. Matching and the inference procedure do not remove confounding from unrecorded conditions. Careful interpretation of performance comparisons under variability is consistent with the methodological concerns raised by Chen et al., 2015.

4.2 Specification and Measurement Alignment:

For each matched pair, the analysis calculates the difference between the two observed coefficients of variation. The average difference is calculated as:

$$\widehat{\Delta} = \frac{1}{S}\sum_{s=1}^{S}\left[\frac{1}{M_s}\sum_{j=1}^{M_s}\left(CV_{sj}^{A} - CV_{sj}^{B}\right)\right] \qquad 2$$

In Equation 2, $\widehat{\Delta}$ is the estimated mean difference in execution time variability; S is the number of servers; $M_s$is the number of matched pairs for server s; and CV_{sj}^{A} and CV_{sj}^{B} are the coefficients of variation for the two configurations in matched pair j. For frequency scaling, Adenotes the enabled setting and Bthe disabled setting. For thread count, Adenotes 20 threads and Bone thread. A positive difference indicates greater variability under configuration A.

The full sample contains 183 servers and 3,294 matched pairs for each comparison. Each server contributes 18 pairs to each comparison: nine benchmarks at each level of the remaining configuration setting. The inner average gives equal weight to matched pairs within a server, and the outer average gives equal weight to servers. This structure prevents a server from receiving greater weight because it has more recorded executions within a group.

Each coefficient of variation is calculated from 3 to 25 executions. The analytical file contains a complete set of observed benchmark and configuration categories for each server, but execution counts and observation spans differ across groups. The sample describes the available CloudLab records and is not assumed

to represent all scientific computing platforms. Laaber et al., 2019, demonstrate that performance variability can differ across benchmarks and environments, supporting the use of workload specific matching.

4.3 Processing, Estimation and Inference:

The analysis uses the cleaned analytical workbook described in the Data section. It retains all 6,588 groups, with no missing values in the variables used for the comparisons. No imputation or outlier removal is applied in the main analysis. Group coefficients of variation were calculated from recorded execution times before the matched comparisons; each coefficient summarizes the executions available for its group.

The procedure first summarizes the distribution of group coefficients and calculates the mean coefficient across the 36 groups for each server. It then forms matched pairs using the relevant server, benchmark and configuration keys. Pair differences are averaged within servers, followed by the calculation of the mean and standard error across the 183 server averages. A two-sided 95 percent confidence interval is calculated as:

$$\widehat{\Delta} \pm t_{0.975,\, S-1} \frac{s_d}{\sqrt{S}} \qquad 3$$

In Equation 3, $t_{0.975,\, S-1}$ is the critical value from a studenttdistribution with $S - 1$degrees of freedom, and $s_d$is the sample standard deviation of the server average differences. The standard error is $s_d$divided by the square root of S. Inference treats the server averages as independent units and allows matched differences within a server to contribute to the same server average. It does not account for possible dependence shared across servers.

The calculations used Python 3.12.14, pandas 2.2.3, NumPy 2.3.5, SciPy 1.17.0 and openpyxl 3.1.5. No random sampling or stochastic optimization was used. Performance comparisons benefit from reporting both estimates and uncertainty because execution times can vary across repeated measurements, as discussed by Irving et al., 2020.

4.4 Sensitivity and Reproducibility:

Two exploratory checks assess whether the main comparisons depend on particular parts of the sample. The first retains a matched pair only when each of its two groups contains at least 10 executions. This check addresses the concern that coefficients calculated from fewer repetitions may be less precise. It changes the analytical sample to 175 servers and 3,150 matched pairs for each comparison. Differences from the main estimates therefore reflect both the restriction and the changed sample.

The second check excludes hp138, the server with the highest mean coefficient of variation across its 36 groups. This check assesses the influence of that observed server on the average comparisons. It does not classify its observations as erroneous. Both checks were conducted after inspection of the data and are interpreted as exploratory rather than prespecified validation.

The confidence intervals do not incorporate a separate estimate of uncertainty in each group coefficient of variation. Unequal execution counts, differing observation spans and possible dependence across servers remain relevant to interpretation. A favourable sensitivity result does not establish causality or show that the estimates transfer unchanged to other computing environments. Duplyakin et al., 2020, document temporal changes in CloudLab performance, highlighting why observation periods matter when interpreting grouped variability measures.

Replication requires the source CPU file, the grouping rules, the analytical workbook and the calculation procedure specified above. The workbook preserves the grouping keys, execution counts, means, sample standard deviations and coefficients of variation needed to reproduce the matched comparisons. The public source file permits reconstruction of the group measures from individual execution records.

5. Findings:

The analysis examines how execution time variability differs across servers, frequency scaling settings and thread counts. It uses matched comparisons to address the three research questions while retaining benchmark identity. The findings describe associations within the observed CloudLab sample and assess their sensitivity to the number of recorded executions.

5.1 Sample Characteristics and Execution Time Variability:

The analytical sample comprises 6,588 server and configuration groups from 183 servers. Each group represents one server, one benchmark, one frequency scaling setting and one thread count. The groups contain between 3 and 25 executions, producing 120,564 recorded execution times. Each server contributes the same 36 configuration combinations, consisting of nine benchmarks, two frequency scaling settings and two thread counts. This structure permits comparisons under common benchmark and configuration categories, although the number and timing of executions vary across groups.

Execution time variability is measured by the coefficient of variation, calculated as the sample standard deviation of execution time divided by its mean and expressed as a percentage. The group mean coefficient is 0.520 percent, the median is 0.373 percent and the standard deviation is 0.605 percentage points. Values range from 0.019 to 8.096 percent. The difference between the mean and median indicates that higher variability in some groups raises the overall mean. The range also shows why a single average cannot describe the full

distribution of benchmark behavior. Irving et al., 2020, similarly emphasize the value of considering the distribution of performance measurements when comparing computer systems.

Average variability differs across the nine benchmarks. The mean coefficient is 0.301 percent for IS and 0.355 percent for BT, compared with 0.640 percent for FT and 0.662 percent for SP. These values describe the observed groups across both frequency scaling settings and thread counts. They do not establish why one benchmark exhibits greater variability than another. The differences nonetheless indicate that workload identity matters when interpreting configuration comparisons. Laaber et al., 2019, also report substantial differences in variability across benchmark and computing conditions.

5.2 Execution Time Variability Across Server Identities:

Average variability across the 36 observed configurations differs among servers. The lowest server mean is 0.336 percent for hp053, while the highest is 2.402 percent for hp138. The median of the 183-server means is 0.462 percent, and the middle half of server means extends from approximately 0.434 to 0.508 percent. The highest mean is therefore well above the center of the observed server distribution. Because each mean covers the same benchmark and configuration categories, the comparison is not driven by differences in the number of categories assigned to each server.

The first research question is answered at the descriptive level: the observed servers differ in their average execution time variability under the common set of configurations. The range does not show that every pair of servers differs statistically, and server identity does not reveal which machine characteristic explains an observed difference. Morgan et al., 2021, document performance variation across computing nodes in parallel scientific calculations, providing related evidence that node identity can matter for observed performance. Their results do not establish the cause of the server differences measured here.

The time dimension also limits interpretation. Group coefficients combine executions recorded over different spans, and the available aggregated workbook retains only the first and last timestamp for each group. A server with a higher coefficient may therefore reflect persistent variation, changes across its observation period, or a combination of these patterns. Duplyakin et al., 2020, document changes in CloudLab performance over time. Their evidence makes temporal coverage relevant to interpretation, but the present group summaries do not identify a temporal change within a particular server.

5.3 Matched Configuration Comparisons:

The analysis compares frequency scaling settings within the same server, benchmark and thread count. It compares thread counts within the same server, benchmark and frequency scaling setting. Each comparison contains 3,294 matched group pairs. Matching holds the recorded comparison categories constant, so the resulting difference concerns the coefficient of variation under two observed configurations within the same server and benchmark.

For each comparison, the matched differences are averaged within each server. The overall estimate is the mean of the 183 server averages, giving each server equal weight. The standard error and 95 percent confidence interval are calculated across those server averages. This approach accounts for the fact that several matched differences belong to the same server, while relying on an assumption that server averages are independent. It does not address possible dependence across servers or establish causal effects. Papadopoulos et al., 2021, emphasize transparent measurement and reporting choices in cloud performance evaluation.

Comparison	Matched Group Pairs	Mean Difference in Variability, Percentage Points	Standard Error	95 Percent Confidence Interval
Frequency scaling enabled compared with disabled	3,294	0.130	0.035	0.061 to 0.199
20 threads compared with one thread	3,294	0.137	0.027	0.083 to 0.191

5.4 Frequency Scaling and Execution Time Variability:

Enabled frequency scaling is associated with an average coefficient of variation 0.130 percentage points higher than the disabled setting. The 95 percent confidence interval extends from 0.061 to 0.199 percentage points. Within the assumptions used to calculate the interval, the comparison supports a positive average difference in the observed configurations. The result concerns relative execution time dispersion rather than the absolute duration of a benchmark.

The server average difference is positive for 165 of the 183 servers. This pattern shows that the overall estimate is not based solely on the direction of a few server averages. It does not mean that every individual matched pair has a positive difference, nor does it imply that enabled frequency scaling has the same association with every workload. The comparison is associational because the available workbook does not establish randomized assignment or measure all operating conditions that could affect execution time.

The benchmark results show this variation. FT has an average enabled versus disabled difference of 0.432 percentage points, while MG has a difference of 0.326 percentage points. CG instead has an average

difference of minus 0.028 percentage points. These are descriptive benchmark comparisons and have not been separately tested here. They answer the second research question by showing a positive overall association alongside a benchmark for which the observed average direction differs. Rauber et al., 2018, find that performance responses to processor frequency depend on application characteristics, which supports interpreting the differences in relation to workload rather than assuming one common response.

5.5 Thread Count and Execution Time Variability:

The comparison of 20 threads with one thread produces an average difference of 0.137 percentage points in the coefficient of variation. Its 95 percent confidence interval extends from 0.083 to 0.191 percentage points. The server average difference is positive for 169 of the 183 servers. Under the stated inference assumptions, the evidence therefore indicates a higher average relative dispersion for the 20-thread configuration within the observed benchmark and frequency scaling combinations.

The overall estimate does not imply that more threads increase variability for every benchmark. FT shows an average difference of 0.653 percentage points and LU shows 0.622 percentage points. By contrast, the average difference for CG is minus 0.173 percentage points. These figures indicate that the aggregate comparison combines workloads with different observed responses. They are not separate estimates of causal effects, and the benchmark data do not identify which execution processes produced the differences.

Akkan et al., 2013, explain that interruptions can affect parallel execution when delayed tasks reach synchronization points. That mechanism provides a possible interpretation of variability in parallel workloads, but the workbook does not measure interruptions or synchronization delays. The third research question is therefore resolved as an observed comparison: the 20 thread setting has a higher average coefficient of variation in the full matched sample, while the benchmark patterns differ.

5.6 Sensitivity of the Configuration Comparisons:

An exploratory sensitivity analysis retained a matched pair only when both configuration groups contained at least 10 executions. This restriction retained 3,150 pairs across 175 servers for each comparison. The mean frequency scaling difference was 0.131 percentage points, with a 95 percent confidence interval from 0.059 to 0.204. The mean thread count difference was 0.138 percentage points, with an interval from 0.083 to 0.194.

Both estimates remain close to their full sample values of 0.130 and 0.137 percentage points. The check indicates that the average patterns are not dependent on groups with fewer than 10 recorded executions. It does not prove that every retained coefficient is equally precise or that a minimum of 10 runs is sufficient for every workload. Irving et al., 2020, show that performance comparisons require attention to uncertainty and distributional variation; the restriction addresses repetition count but does not resolve all sources of measurement uncertainty.

A second exploratory check excluded hp138, the server with the highest overall mean variability. The frequency scaling difference became 0.152 percentage points, with a 95 percent confidence interval from 0.098 to 0.206. The thread count difference became 0.154 percentage points, with an interval from 0.113 to 0.196. Both estimates remained positive and were somewhat larger than the full sample estimates. The observed average patterns therefore do not depend on including this server, although the check does not establish whether its high variability reflects an error or a valid feature of its recorded executions.

The three research questions yield a consistent but qualified account. Server means differ within the common set of configurations. Enabled frequency scaling and the 20-thread setting are each associated with higher average execution time variability in matched comparisons. The benchmark summaries show that the direction and magnitude of those differences are not uniform. These findings concern dispersion in execution time within the available CloudLab records; they do not establish numerical reproducibility failures or identify physical causes.

6. Discussion:

Stable computational results depend on more than selecting a server and reporting its average execution time. The matched comparisons indicate that enabled frequency scaling and the 20-thread configuration are each associated with higher average execution time variability, by 0.130 and 0.137 percentage points, respectively. Both confidence intervals exclude zero under the stated server independence assumption. Equation 1 defines variability relative to mean execution time, so these differences concern consistency rather than speed. The contribution of this analysis is the comparison of both settings within the same observed servers and benchmarks, alongside differences among server identities. It adds a configuration focused perspective to performance comparison, where Chen et al., 2015, show that ignoring variability can distort conclusions about computing systems. The associations do not establish that either setting causes the observed differences.

The benchmark patterns qualify any interpretation of a common configuration effect. The frequency scaling comparison is positive on average, but the observed difference is slightly negative for CG. The thread comparison is positive overall and particularly large for FT and LU, while CG shows an average decrease. These contrasts suggest that workload characteristics are relevant to how configuration settings are associated with execution time dispersion. Zhang et al., 2017, examine variability in parallel workloads and show why

interactions among execution conditions merit attention. The present analysis makes those differences visible at the benchmark level, but it does not measure the resource contention, scheduling behavior or processor frequency changes needed to identify the mechanisms. The benchmark patterns are descriptive and should not be interpreted as established mediation pathways.

The server results add a second dimension to the interpretation: the same set of observed benchmark and configuration categories produces different average variability across machines. This finding cautions against treating a single server as representative of every server in the sample. It also suggests that an overall configuration estimate can coexist with substantial differences in machine behavior. Pusukuri et al., 2012, demonstrate that scheduling and memory allocation policies can affect variation in repeated executions. Those policies were not measured in the available workbook, so they remain possible explanations rather than demonstrated channels for the server differences. The analysis therefore identifies where variability differs, while leaving the physical and operational sources of that difference to direct measurement.

The sensitivity results strengthen the interpretation of the average comparisons without eliminating their boundaries. Requiring at least 10 executions in both groups of a matched pair leaves the two estimates close to their full sample values, and excluding the server with the highest overall mean variability preserves their positive direction. These checks reduce concern that either result depends solely on the shortest recorded groups or one high variability server. They do not address shared disturbances across servers, changes within observation periods or unmeasured configuration conditions. The distinction matters for international computing research because Leitner and Cito, 2016, identify provider and shared environment differences in public cloud performance, whereas the present evidence comes from a specific CloudLab server sample. The two settings raise related questions about variability, but their magnitudes and sources should not be assumed equivalent.

For researchers and computing operators, the practical implication is to compare repeated benchmark results within the same workload and report both average execution time and its dispersion. The contrasting FT, LU and CG patterns argue against selecting a frequency scaling or thread setting solely from an overall average. A useful next step is to repeat matched comparisons across additional server platforms while recording actual processor frequencies, scheduling events and execution timestamps. Such measurements could test whether the observed benchmark differences arise from identifiable operating mechanisms and whether the associations transfer to other infrastructures. Iosup et al., 2011, show that variability matters across production cloud services; extending the present comparison to multiple environments would test its broader relevance without assuming that one facility represents the global computing landscape.

7. Conclusion and Implications:

Reliable scientific computing requires attention to the consistency of performance as well as its speed. The analysis shows that execution conditions are associated with variability, while the direction and magnitude of observed differences depend on the workload. This evidence adds a matched comparison of settings within servers and benchmarks to the evaluation of computational performance. It suggests that theories of performance variability should account for differences among workloads rather than assume a uniform response to configuration choices. For computing operators, the findings support comparing repeated executions under matched conditions before selecting settings for scientific workloads. Research institutions can strengthen evaluation protocols by recording execution conditions and reporting dispersion alongside average performance. Such practices can help researchers distinguish operational variation from meaningful performance differences and make computational evidence easier to assess across institutions. The results are relevant to international benchmarking practice, although their transfer to other platforms requires further evaluation.

The analysis is limited to one computing environment and measures associations rather than causal mechanisms. The aggregated records do not identify the precise sources of variation or preserve the timing of every execution. Future research can repeat the comparisons across platforms using individual execution records, direct measurements of processor frequency and scheduling activity, and controlled configuration changes. These extensions would help determine which mechanisms explain the observed patterns and whether the findings generalize to other scientific workloads.

References:

1. Akkan, H., Lang, M., & Liebrock, L. 2013. Understanding and isolating the noise in the Linux kernel. The International Journal of High Performance Computing Applications, 27, issue 2, 136-146. https://doi.org/10.1177/1094342013477892
2. Chen, T., Guo, Q., Temam, O., Wu, Y., Bao, Y., Xu, Z., & Chen, Y. 2015. Statistical performance comparisons of computers. IEEE Transactions on Computers, 64, issue 5, 1442-1455. https://doi.org/10.1109/TC.2014.2315614
3. Duplyakin, D. 2020. Benchmarking data collected on CloudLab. Dataset, version 6. Zenodo. https://doi.org/10.5281/zenodo.3686952
4. Duplyakin, D., Uta, A., Maricq, A., & Ricci, R. 2020. In datacenter performance, the only constant is change. Proceedings of the 20th IEEE and ACM International Symposium on Cluster, Cloud and Internet Computing, 370-379. IEEE. https://doi.org/10.1109/CCGrid49817.2020.00-56

5. Iosup, A., Yigitbasi, N., &Epema, D. H. J. 2011. On the performance variability of production cloud services. Proceedings of the 11th IEEE and ACM International Symposium on Cluster, Cloud and Grid Computing, 104-113. IEEE. https://doi.org/10.1109/CCGrid.2011.22
6. Irving, S., Li, B., Chen, S., Peng, L., Zhang, W., & Duan, L. 2020. Computer comparisons in the presence of performance variation. Frontiers of Computer Science, 14, issue 1, 21-41. https://doi.org/10.1007/s11704-018-7319-2
7. Laaber, C., Scheuner, J., & Leitner, P. 2019. Software microbenchmarking in the cloud. How bad is it really? Empirical Software Engineering, 24, issue 4, 2469-2508. https://doi.org/10.1007/s10664-019-09681-1
8. Leitner, P., & Cito, J. 2016. Patterns in the chaos: A study of performance variation and predictability in public IaaS clouds. ACM Transactions on Internet Technology, 16, issue 3, article 15, 1-23. https://doi.org/10.1145/2885497
9. Morgan, H., Sanan, P., Knepley, M., & Mills, R. T. 2021. Understanding performance variability in standard and pipelined parallel Krylov solvers. The International Journal of High Performance Computing Applications, 35, issue 1, 47-59. https://doi.org/10.1177/1094342020966835
10. Papadopoulos, A. V., Versluis, L., Bauer, A., Herbst, N., von Kistowski, J., Ali-Eldin, A., Abad, C. L., Amaral, J. N., Tůma, P., & Iosup, A. 2021. Methodological principles for reproducible performance evaluation in cloud computing. IEEE Transactions on Software Engineering, 47, issue 8, 1528-1543. https://doi.org/10.1109/TSE.2019.2927908
11. Pusukuri, K. K., Gupta, R., & Bhuyan, L. N. 2012. Thread tranquilizer: Dynamically reducing performance variation. ACM Transactions on Architecture and Code Optimization, 8, issue 4, article 46, 1-21. https://doi.org/10.1145/2086696.2086725
12. Rauber, T., Rünger, G., & Stachowski, M. 2018. Performance and energy metrics for multi-threaded applications on DVFS processors. Sustainable Computing: Informatics and Systems, 17, 55-68. https://doi.org/10.1016/j.suscom.2017.10.015
13. Zhang, W., Ji, X., Song, B., Yu, S., Chen, H., Li, T., Yew, P. C., & Zhao, W. 2017. VarCatcher: A framework for tackling performance variability of parallel workloads on multi-core. IEEE Transactions on Parallel and Distributed Systems, 28, issue 4, 1215-1228. https://doi.org/10.1109/TPDS.2016.2613524