Role of AI in SoC Performance Verification(PV)

Sharada Vajja, Raghu Alamuri, Saksham Mehra

GOOGLE LLC

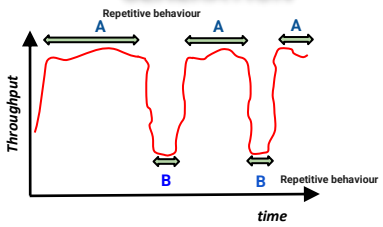
accellera SYSTEMS INITIATIVE

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA MARCH 4-7, 2024

PROBLEM STATEMENT

P1. STIMULUS GENERATION



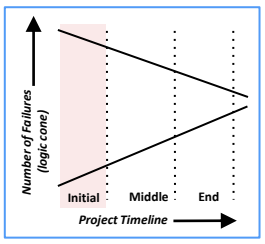
Ultimate goal of PV is to analyze Usecases / Benchmarks in Pre-Si simulation/emulation environment. The effort also aids in competitive analysis of products.

Long running usecase / benchmark are critical for testing SOCs.

Takes days to run on simulation.

Many parts of the usecase show repetitive behaviour. Is it necessary to run everything ??

P2. PERFORMANCE FAILURES



- Single failure signature in performance tests →

Cycle count mismatch

Too many failures to debug ....

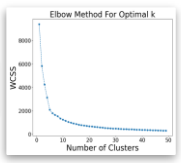
How to prioritize ??

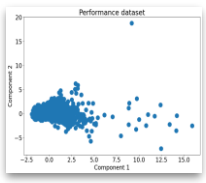
AI to rescue....

K-Means Clustering

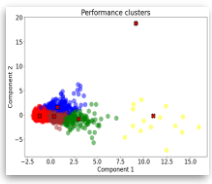
- K-means is an unsupervised machine learning algorithm that groups input data into distinct clusters.
- The value of k i.e the number of clusters is an input to the algorithm and needs to be determined empirically.
- Elbow method is used to find the optimum value of k.
- The method involves plotting the within cluster sum of square scores(WCSS) with varying number of clusters.
- The point of elbow on this plot is taken to be as the optimum value of k.

WCSS = $\sum_{i=1}^{N_C} \sum_{x \in C_i} d(\mathbf{x}, \bar{\mathbf{x}}_{C_i})^2$





Dataset before K-Means



Clustered dataset after K-Means

Principal Component Analysis

- Principal component analysis (PCA) is a statistical procedure that reduces the dimensionality of large datasets, while preserving crucial information.
- Having too many features/dimensions increases the risk of overfitting.
- Dimensionality reduction helps in identifying and keeping only the most informative features from a dataset with too many features.

PC_1 = a_11 * x_1 + a_12 * x_2 + ... + a_1p * x_p

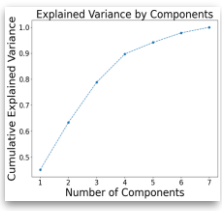
PC_2 = a_21 * x_1 + a_22 * x_2 + ... + a_2p * x_p

...

k <= p

PC_k = a_k1 * x_1 + a_k2 * x_2 + ... + a_kp * x_p

a_ij represents original dimensions of the dataset. PC_ij represents the principal components generated from original dimensions. Number of principal components can be less than or equal to original number of dimensions.



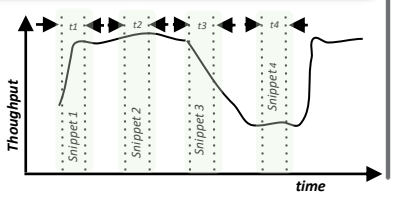
The number of PCA components which can represent upto 80% cumulative variance of the original data, is taken to be the optimum number.

Intelligent Stimulus Generation

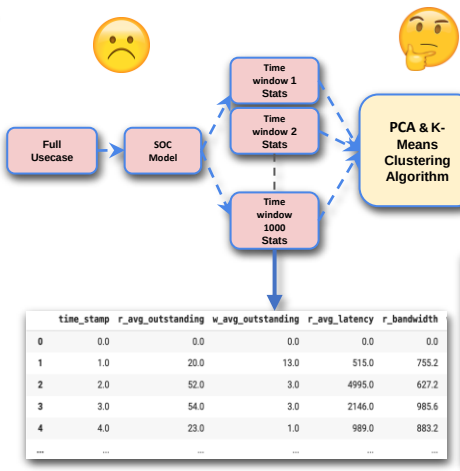
Motivation	AI Solution	PCA + K-Means	Algorithm's output
• Many parts of a long running usecase show statistically similar behaviour. • We need a mechanism which can identify this similarity and give us a shorter trace(address access patterns) which represents the entire usecase behaviour.	• AI comes to our rescue !! • We break the complete usecase into time windows of equal size and collect a number of statistics for each frame. • Each window is now a data point for the algorithm, with the collected stats as its features/dimensions.	• We then extract the principal components from the dataset using PCA. • Elbow method is used to determine optimum k, and the reduced dimensionality dataset is fed into the k-means model.	• K-Means clusters the dataset into k groups and gives as output the cluster centres. • These cluster centres represent the behaviour of all other time snippets falling in the cluster. • We can now run on simulation, only the address traces corresponding to windows at the cluster centres.

Average Bandwidth	Average outstanding requests	DRAM Page Hits	DRAM Cache Hits
Average Latency	DRAM Refresh Counts	DRAM Activates	MMU Hits

Sample statistics captured for every time window in a usecase.

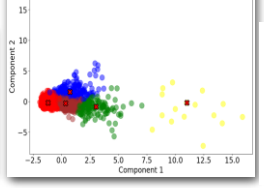


Intelligent Stimulus Generation

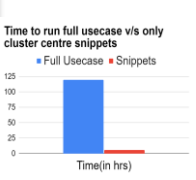


	time_stamp	r_avg_outstanding	w_avg_outstanding	r_avg_latency	r_bandwidth
0	0.0	0.0	0.0	0.0	0.0
1	1.0	20.0	13.0	515.0	755.2
2	2.0	52.0	3.0	4995.0	627.2
3	3.0	54.0	3.0	2146.0	985.6
4	4.0	23.0	1.0	989.0	883.2
...	...	...	...	...	...

Each row of the table represents the stats captured corresponding to a time window in the complete usecase



Each point on the plot represents a snippet(time window) from full usecase run (plotted in terms of the PCA components calculated from window stats). Each color represents a cluster and each cross is the corresponding cluster centre.



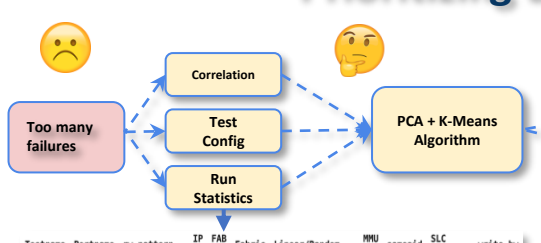
Prioritizing Debugs

Motivation	AI Solution	PCA + K-Means	Algorithm's output
• Unlike functional simulations, performance verification failures often lack unique and self descriptive error signatures, showing up as cycle count mismatch or throughput discrepancies. • Notably, 90% of failures arise from common bottlenecks falling into a "Big-Bucket" category.	• We mathematically model each performance failure using various statistics collected during runtime along with RTL configuration settings. • Each of the failing test thus becomes a datapoint, with collected stats as features/dimensions.	• We then extract the principal components from the dataset using PCA. • Elbow method is used to determine optimum k, and the reduced dimensionality dataset is fed into the k-means model.	• K-Means clusters the dataset into k groups and gives as output the cluster centres. • We can then begin our debugs with the cluster centres, which in many cases is representative of bottlenecks for all tests in a cluster, eventually saving time. • Analyzing each cluster's properties, also gives us insights into probable cause of failures for those cluster members.

DDR Activate count	write_average_outstanding (wr_avg_os)	IP Max Outstanding (IP MO)	Fixed AxID (sameaid)	Address pattern (Linear/Random)
DDR Refresh count	Memory Management Unit Enable/Disable (MMU Enable)	Fabric Max Outstanding (FAB MO)	Cache hit enable/disable (SLC Hit)	Virtual Channel targeted (FAB VC)

Sample statistics/configuration settings captured for every failing test.

Prioritizing Debugs

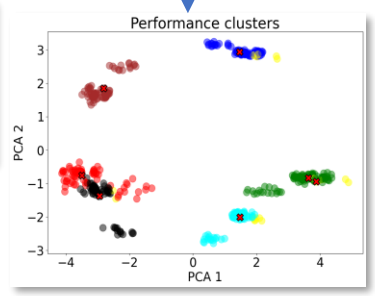


We can further analyze each cluster's properties (for e.g. the mode of stat/config within the cluster members) to infer probable cause of failures

Testname	Portname	rw_pattern	IP MO	FAB MO	Fabric	Linear/Random	MMU Enable	sameaid	SLC Hit	...	write_bw
Test0	Port0	1.0	48.0	48	100.0	0.0	0.0	0.0	0.0	...	0.026569
Test1	Port1	1.0	64.0	64	100.0	0.0	0.0	0.0	0.0	...	0.044207
Test2	Port2	2.0	32.0	32	100.0	0.0	0.0	0.0	0.0	...	0.000000
Test3	Port3	2.0	160.0	160	100.0	0.0	0.0	0.0	0.0	...	0.000000
Test4	Port4	1.0	256.0	256	100.0	0.0	0.0	0.0	0.0	...	0.162033
...	...	...	...	...	...	...	...	...	...	...	...

Each row is a failing test along with the run statistics and configuration settings for that.

All failures are divided into buckets(clusters) and we can start debugging with few tests(or cluster's centroid) from the bucket. Most tests in a bucket fail due to same bottlenecks, thus reducing debug effort.



Each point here is a failing test mathematically modelled in terms of its run statistics/configs. Each colour is a cluster of failing tests, out of which we begin debugs with only the centroids(marked with crosses).

Google

Authors' contacts : msaksham, raghualamuri, sharadav @google.com