Workload Pruning for Effective Architecture Exploration

Byeong Kil Lee

Abstract: ***Design exploration requires the detailed simulation which is running multiple applications on a cycle-level microprocessor simulator. Main objectives of simulation-level design exploration include understanding the architectural behaviors of target applications and finding optimal configurations to cover wide range of applications in terms of performance and power. However, full simulation of an industry standard benchmark suite takes several weeks to months to complete. This problem has motivated several research groups to come up with methodologies to reduce simulation time while maintaining a certain level of accuracy. Among many techniques for reducing simulation time, a tool called SimPoint is popularly used. However, simulation load even with the reduced workloads is still heavy, considering design complexity of modern microprocessors. Motivation of this research is started from how design exploration is actually performed. Designers will observe the performance impact from resource variations or configuration changes. If a simulation point shows low sensitivity to resource variations, designers would skip those simulations. In this paper, we focus on identifying those simulation points which do not give big impact to representative behaviors, by which overall simulation time can be effectively reduced. We also performed the performance-sensitivity-based similarity analysis (K-mean clustering) among simulation points on specific performance metric which can lead to effective workload pruning.***

Index: ***Workload Characterization; Performance Evaluation; Workload Reduction; Early-Stage Design Exploration; Performance Evaluation.***

I. INTRODUCTION

Design exploration requires the detailed simulation which is running applications on a cycle-level microprocessor simulator. Simulators are extremely valuable tool for computer architects which can reduce the cost and time of a project by allowing the architect to quickly evaluate different processor implementations. Additionally, they allow the architect to quickly determine the expected performance improvement of a new processor enhancement [1]. However, full simulation of an industry standard benchmark suite (e.g., SPEC CPU 2006 [6]) takes several weeks to months to complete. This problem has motivated several research groups to come up with methodologies to reduce simulation time while maintaining a certain level of accuracy.

One of the popular techniques for reducing simulation time, a tool called SimPoint [10][11] is popularly used. The SimPoint employs offline phase classification algorithm which calculates phases for a program/input pair, and then chooses a single representative from each phase and estimates the remaining intervals. The tool chooses this representative for each phase by finding the interval closest to the cluster's centroid.

This selected interval for a phase is called a simulation point for that phase. Then, detailed simulations are performed at the simulation points and weigh each performance metric values by the size in its cluster. In addition to multiple simulation points, SimPoint also provides the mechanism to get both standard single simulation point and early single simulation point for each benchmark [11].

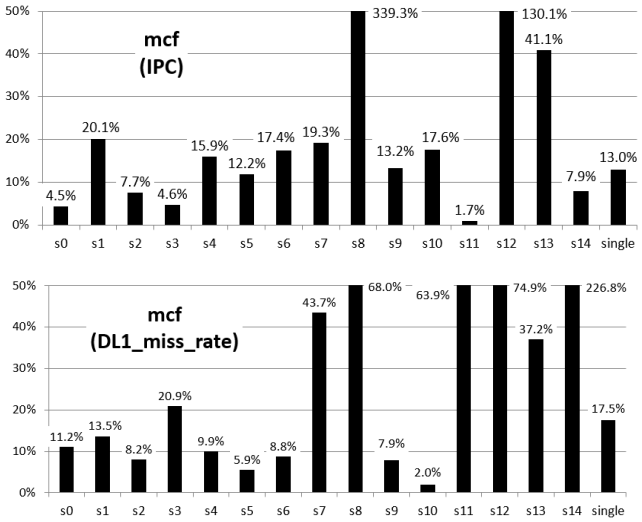


Fig. 1. Error rate of IPC and cache miss rate: individual simulation points vs. single simulation point. (reference: full simulation)

Even with the reduced simulation workloads, design exploration time cannot be ignored. Particularly in MID (mobile internet devices) domain, performance exploration and evaluation time is very critical. A standard single simulation point, which is extracted from the Sim Point tool, can be a solution for reducing the simulation time. However, as shown in Figure 1, a standard single simulation point does not provide accuracy. Figure 1 shows the percentage error to the full simulation with respect to IPC and cache miss rate. In the case of IPC, some simulation points such as s0, s2, s3, s5, and s11 show better accuracy than single simulation point (right-most one). Particularly, s11 shows the smallest difference (1.7%) while s8 shows the biggest difference (339.3%). Each individual point has its weight information (from the SimPoint tool) which is used for overall metric calculation with multiple simulation points. Fortunately, s8 (0.2) and s11 (0.6) has small weights which means their impact to multiple simulation points is not remarkable.

Revised Version Manuscript Received on March 05, 2019.

Byeong Kil Lee, Department of Electronics and Communication Engineering, UCCS, 1420 Austin Bluffs Parkway, Colorado Springs, CO 80918, USA.

This study is based on above observations and two fundamental motivations: (i) pruning of simulation points instead of using all simulation points or a single simulation point; and (ii) identifying the simulation points which are less-sensitive to resource variation because it might be a waste of time to evaluate performance without sensitive variation on resource changes or hardware configuration changes.

In this paper, we propose a performance-sensitivity-based workload pruning mechanism for improving simulation methodology. From the above example in Figure 1, the overall IPC value from all simulation points with weight information shows 12.1% error rate to full simulation. Performance evaluation with standard single simulation point is closed to the result with multiple simulation points, but it is not the best choice for all metrics. Also, some phase of simulation points in dynamic simulation constraint overall performance. We want to extract this critical information through the performance-sensitivity-based workload pruning. The proposed mechanism can also apply to investigate fine-grained similarity in inter-application level. This information can be useful for multi-threading and multi-core simulation.

Basic motivation of this research is started from how design exploration is actually performed. Designers will observe the performance impact from resource variations or configuration changes. If a simulation point shows low sensitivity to resource variations, designers would eliminate those simulation points from the simulation setup procedure. This research is a follow-up study of Yi's research [1], Simpoints methodology [10] and Raghunath's approach [23].

The rest of the paper is organized as follows. In Section II, we describe statistical approaches such as Simpoints, Plackett and Burman designs and resource boundaries for hardware components. The proposed research is described in Section III which gives details about PB design matrix with simulation points, ranking and performance-sensitivity-based grouping methodology, and discusses the degree of performance sensitivity, the degree of performance similarity and validity check of the proposed pruning scheme. In Section IV, we describe the related work, and we conclude with section V.

II. STATISTICAL APPROACH

1.1. Simulation Points

The full simulation of SPEC 2006 benchmarks takes long time because of large number of instructions and large number of data access footprints [8]. Hence, it is very difficult to conduct the performance and power estimation of such application benchmarks at each stage of the design. It will be getting worse as the complexity of microprocessors keeps increasing. Simpoint methodology [10] is proposed to extract sets of simulation points for the general-purpose benchmark suite [8]. Simpoint method is used for capturing and separating unique phase behavior that exists in many programs. In our experiment, first, we extract the basicblock vectors with an interval of 100,000,000 instructions, and then the clustering algorithm is applied. The maximum value of k (maxK) is taken as 30. Simplescalar's [19] fastfwd functionality is used to simulate each simpoints, and the various metric values are obtained for each simpoint. With maxK value of 1, we extract a standard single simulation point.

1.2. Plackett and Burman Designs

The Plackett and Burman (PB) design has been applied in Yi's research [1] to investigate statistical similarity among SPEC 2000 benchmarks with the reduced input sets [13]. PB design was chosen due to its relatively fewer simulations required, compared to other methods [3]. The PB design with N parameters requires (N+1) simulations which is minimal number of simulations required to estimate the effect of each of the N parameters. There is an improvement on the original PB design call "foldover" PB design [5]. It requires approximately 2N simulations. Because PB design exist only in sizes that are multiples of 4, the base PB design requires X simulation combination cases, where X is the next multiple of 4 that is larger than N, and the foldover PB design requires 2X simulations.

In our study, we applied the PB design to investigate how resource parameters impact on the processor's performance from fine-grained simulation point level rather than application level. We picked similar hardware parameters and PB values used in Yi's paper [1], but we use 31 PB design parameters which end up with 64 distinct configurations. As PB design does not simulate every possible combination of cases, it should be noted that it cannot quantify the effects of all of the interactions. However, fortunately, the results in [4] show that if an interaction between parameters is critical, the result will be meaningful only because each of the constituent parameters are equally important to the result. Hence, applying foldover to PB design does not compromise the results.

The parameter's evaluation for each simulation is given by PB matrix. The matrix size of foldover PB design will be 2X*(X-1). When N < (X-1), extra columns are redundant which have no effect on simulation results. The value of the matrix's first row is given by [2], and the next X-2 rows are formed by doing a circular right shift on the preceding row. The last row is a row of minus ones. The foldover part is exactly the invert values of the upper part [1].

Table1. Plackett-Burman Design in 12 Runs for up to 11 Factors

	Pattern	N1	N2	N3	N4	N5	N6	N7	N8	N9	N 10	N 11
1	++-+++---+-	1	1	-1	1	1	1	-1	-1	-1	1	-1
2	-++-+++---+	-1	1	1	-1	1	1	1	-1	-1	-1	1
3	+-++-+++---	1	-1	1	1	-1	1	1	1	-1	-1	-1
4	-+-++-+++--	-1	1	-1	1	1	-1	1	1	1	-1	-1
5	--+-++-+++-	-1	-1	1	-1	1	1	-1	1	1	1	-1
6	---+-++-+++	-1	-1	-1	1	-1	1	1	-1	1	1	1
7	+---+-++-++	1	-1	-1	-1	1	-1	1	1	-1	1	1
8	++---+-++-+	1	1	-1	-1	-1	1	-1	1	1	-1	1
9	+++---+-++-	1	1	1	-1	-1	-1	1	-1	1	1	-1
10	-+++---+-++	-1	1	1	1	-1	-1	-1	1	-1	1	1
11	+-+++---+-+	1	-1	1	1	1	-1	-1	-1	1	-1	1
12	----------	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Table 1 shows a sample PB matrix. For each pattern of configuration, specific performance metrics will be evaluated. The impact to the performance of each parameter can be measured by performance impact values and PB design parameters. Its calculation formula is shown as below.

$$Perf\_Impact_k = \sum_i (\pm 1)_i \times (Perf\_metric)_i$$

where k is N1, N2, N3, …, N11; i is number of rows.

The value "+1" and "-1" describe the hardware configuration for a simulation. Only two values were chosen to represent two extreme cases: "+1" means the parameter's configuration is higher than normal cases, while "-1" means lower than normal situation. In the case of memory configuration, if we set the normal value range for il1 (instruction level-1) cache size is 4KB~128KB, then we can set "+1" =128KB and "-1"=4KB. To conclude, "+1" represents a configuration that guarantee a higher performance, while "-1" means a lower one. This value does not represent numerical value only; it is also used in other parameters such as branch prediction. We can set "+1" = "perfect" and "-1" = "taken (or nottaken)".

Using the Perf_Impact formula, we can calculate the intensity that a hardware parameter impacts on the processor's performance. By examining the magnitude of the impact value, performance-sensitivity (PS) and PS-based similarity can be identified. The sign of the Perf_Impact value has no meaning. In the case of Table 1, final PB matrix size will be 24x11 by applying the "foldover" concept [5].

1.3. Resource Boundaries for Hardware Components

In our experiments, we choose 31 variables for hardware components to avoid "dummy parameters". In other words, PB matrix (X=32) is saturated with 31 variables. Table 2 shows 31 elements, but final matrix size will be 64x31 including the foldover. Eventually, 64 independent hardware configurations will be used for simulations for similarity analysis and sensitivity analysis.

We used similar PB boundary values with Yi's research [1]. In order to choose parameters which can well represent the processor's performance, all aspects of the processor should be taking into account, including processor core parameters, functional unit parameters and memory related parameters. We also need to notice that the setting of the normal value range will also greatly influence the simulation result. Too wide range will inflate the importance of the parameter, while too narrow range has opposite effect. We deliberately choose parameter values to be slightly lower and slightly higher than normal values. Besides, some parameters interact with each other, thus their values cannot be chosen totally independently of other parameters. For example, the inter memory access latency must be much smaller than the first memory access latency. Also, to make the number of parameters to 31 (32 minus 1), we exclude D-TLB page size and latency in the parameters. If we include those two parameters, the number of parameters should be 35 (36 minus 1) – four multiple number, including 2 dummy parameters. Table 2 shows the selections of parameters and their configuration of low value and high value, and Table 3 includes the fixed parameters with default value.

III. PS (PERFORMANCE SENSITIVITY) SIMILARITY-BASED WORKLOAD PRUNING

3.1. PB Design Matrix with Simulation Points

With the designed PB matrix (64x31), we choose six SPEC CPU 2006 benchmarks, soplex, hmmer, astar, perlbench, namd, bwaves which have 11, 9, 15, 16, 24 and 16 simulation points respectively. Each simulation points have been measured with all 64 configurations. In our experiments, we ran 5824 (704+576+960+1024+1536+1024) simulations with Simplescalar toolsets. We use the CPI (cycle per instruction) as an overall performance metric to make rankings on impact among hardware components and simulation points. If the designers want to focus on a specific hardware module (e.g., cache memory, branch prediction module, etc.) for their design exploration, they can choose an individual performance metric to represent each component such as cache miss rate or branch prediction miss rate.

In Yi's paper [1], they use the execution time to represent overall performance, but measuring execution time might cause a lot of possible errors due to other factors such as system time and other machine activity factors by other users. Particularly, we use multiple machines to simulate which will make it worse. Therefore, we choose the CPI as an overall performance metric which can be a pure reference only for the simulation itself.

Table 2. Processor parameters (31) and their PB values

Parameter	Low Value	High Value
Branch predictor	taken	perfect
BTB Entries	16	512
BTB Assoc	2-way	Fully-Assoc
Fetch Queue Entries	4	32
RUU Size	8	64
LSQ Entries	0.25*RUU = 2	1.0*RUU = 64
Memory Ports	1	4
Int ALUs	1	4
Int Mult/Div Units	1	4
FP ALUs	1	4
FP Mult/Div Units	1	4
L1 I-Cache Size	4 KB	128 KB
L1 I-Cache Block Size	16 Bytes	64 Bytes
L1 I-Cache Assoc	1-way	8-way
L1 I-Cache Latency	4 Cycles	1 Cycles
L1 D-Cache Size	4 KB	128 KB
L1 D-Cache Block Size	16 Bytes	64 Bytes
L1 D-Cache Assoc	1-way	8-way
L1 D-Cache Latency	4 Cycles	1 Cycles
L2 Cache Size	256 KB	8192 KB
L2 Cache Block Size	64 Bytes	256 Bytes
L2 Cache Assoc	1-way	8-way
L2 Cache Latency	20 Cycles	5 Cycles
Memory Latency, First	200 Cycles	50 Cycles
Memory Bandwidth	4 Bytes	32 Bytes
I-TLB Entries	32	128
I-TLB Page Size	8 KB	4096 KB
I-TLB Assoc	2-way	Fully-Assoc
I-TLB Latency	80 Cycles	30 Cycles
D-TLB Entries	32	128
D-TLB Page Size	*Same as I-TLB Page Size*	
D-TLB Assoc	2-way	Fully-Assoc
D-TLB Latency	*Same as I-TLB Latency*	

Table 3.Fixed parameters with default value

Parameters	Default Value
Fetch Mis-prediction Latency	3
Fetch Speed	1
RAS Entries	8
Decode Width	4
Issue Width	4
Commit Width	4
L2 I-Cache Latency	6
Spec Branch Update	Non-spec
Issue:inorder	False
Issue:wrongpath	True
Cache:flush	False

Table 4. PB Design Results for all Processor Parameters (ASTAR)

Parameter	sim0	sim1	sim2	sim3	sim4	sim5	sim6	sim7	sim8	sim9	sim10	sim11	sim12	sim13	sim14	Sum
I-TLB Page Size	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**15**
I-TLB Latency	2	2	2	2	2	2	2	2	2	2	3	2	2	2	2	**31**
I-TLB Size	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3	**44**
RUU Size	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	**60**
D-TLB Assoc	7	7	5	8	5	6	7	6	6	5	5	7	7	5	5	**91**
Mem Latency, First	6	12	6	10	10	8	31	8	9	9	9	6	6	13	6	**149**

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Branch predictor	5	31	7	7	12	20	14	5	16	11	16	5	5	8	7	**169**
L2 Cache Size	9	15	11	9	24	14	13	9	8	16	6	9	9	27	16	**195**
LSQ Entries	10	21	23	16	8	9	17	10	11	8	18	10	10	7	25	**203**
L2 Cache Block Size	8	6	25	5	27	21	9	7	5	26	8	8	8	22	22	**207**
L1 D-Cache Size	14	8	10	15	15	12	29	15	18	12	13	14	14	12	10	**211**
Memory Bandwidth	30	5	8	13	6	5	5	30	13	6	21	31	30	6	8	**217**
Fetch Queue Entries	25	17	12	24	7	7	8	21	21	7	7	25	25	9	11	**226**
L2 Cache Assoc	22	11	20	20	9	11	6	19	10	13	11	22	22	11	23	**230**
FP Mult/Div Units	19	9	9	21	13	13	20	22	7	21	26	16	17	30	9	**252**
BTB Assoc	16	27	16	19	22	16	12	16	12	17	10	17	16	21	17	**254**
L1 D-Cache Block Size	13	29	21	23	11	10	23	13	20	10	27	13	12	10	21	**256**
I-TLB Assoc	20	13	29	11	14	15	19	17	19	15	12	21	20	15	31	**271**
L1 D-Cache Assoc	17	14	13	27	25	22	10	23	14	24	29	18	19	24	13	**292**
Int Mult/Div Units	11	18	19	17	26	30	25	12	15	27	31	12	11	23	19	**296**
FP ALUs	21	16	22	22	16	17	26	20	25	14	23	20	21	14	20	**297**
Memory Ports	12	24	14	31	23	23	24	11	24	25	24	11	13	26	14	**299**
L1 I-Cache Assoc	15	25	27	12	21	26	22	14	22	22	25	15	15	17	28	**306**
L1 I-Cache Size	18	26	26	14	20	19	18	18	23	18	28	19	18	16	26	**307**
L2 cache latency	23	19	30	18	17	18	30	24	28	19	15	23	23	18	29	**334**
Int ALUs	24	20	28	25	18	25	11	25	29	23	19	24	24	20	24	**339**
BTB Entries	28	23	18	28	19	24	21	29	30	20	17	28	28	19	18	**350**

L1 I-Cache Block Size	29	10	15	30	31	29	15	28	27	29	14	29	29	28	12	**355**
L1 I-Cache Latency	31	22	24	6	29	27	27	31	17	28	22	30	31	29	30	**384**
D-TLB Size	26	28	17	29	28	31	16	26	31	30	30	26	27	31	15	**391**
L1 D-Cache Latency	27	30	31	26	30	28	28	27	26	31	20	27	26	25	27	**409**

Table 4 shows PB design results (X=32) for all 31 processor parameters which are based on all simulations and ranking by performance impact, where all individual ranks are added for each hardware component (right-most column). The results are based on 64 (=2X) simulations with different hardware configurations for each simulation point. The smaller value means the hardware component is more significant in overall performance (CPI in this experiment). From the result of Table 4, we observe that only the first 7 parameters are significant across all simulation points. This is drawn by examining the difference of the result between the seventh parameter (Branch predictor) and the eighth one (L2 cache size). Based on the PB design results, memory related components such as I-TLB page size/latency/size, D-TLB assoc are potential enhancement factors that can be made to greatly improve the system's performance. The performance impact of each parameter can be clearly seen in this result. It can be really useful for design exploration to observe the influence of certain simulation point or certain parameter.

3.2. Degree of Performance Sensitivity

Microprocessor designers need sets of workloads for their design exploration to decide optimal configuration. The choice of workloads is very critical for their design optimization. Due to long simulation time, designers tend to choose a sample of simulation points or they can use simulation points from Simpoint mechanism [10]. If the selected simulation points are not sensitive to hardware configuration and resource variation, it might be waste of time in design exploration. Therefore, the sensitivity of each simulation point is one of the important factors to choose a benchmark and it needs to be identified for effective design exploration. Based on our experiments with PB design matrix, we can also extract performance sensitivity of each simulation point against 64 different configurations which have sets of resource variations. Figure 2 shows the performance sensitivity for each simulation points. We use the performance sensitivity equation as shown below, for which standard deviation and mean value of the CPI are used to evaluate for each simulation point.

$$performance\ sensitivity(\%) = \frac{std.Dev}{mean} \times 100\%$$

The standard deviation is based on its mean value, but the mean values for each simulation points are widely spread. In this equation, standard deviation value is divided by each mean value observe each simulation point's performance sensitivity. As shown in Figure 2 (a), the soplex simulation points 0, 3 and 9 are identified as high-sensitive simulation points for overall performance. These higher sensitivity groups (0, 3 and 9) should be independently simulated and should not be categorized into any similarity group. We can observe that all other groups have similar performance sensitivity within 10~15% boundary. The hmmer simulation results show very similar sensitivity among simulation points as shown in Figure 2 (b). The simulation point 0 and 3 are distinct (relatively less sensitive), and all other simulation points are very similar. The hmmer can be categorized as very high performance-sensitivity group in both simulation point level and application level, but it doesn't need to simulate all simulation points due to similar performance sensitivity among them.

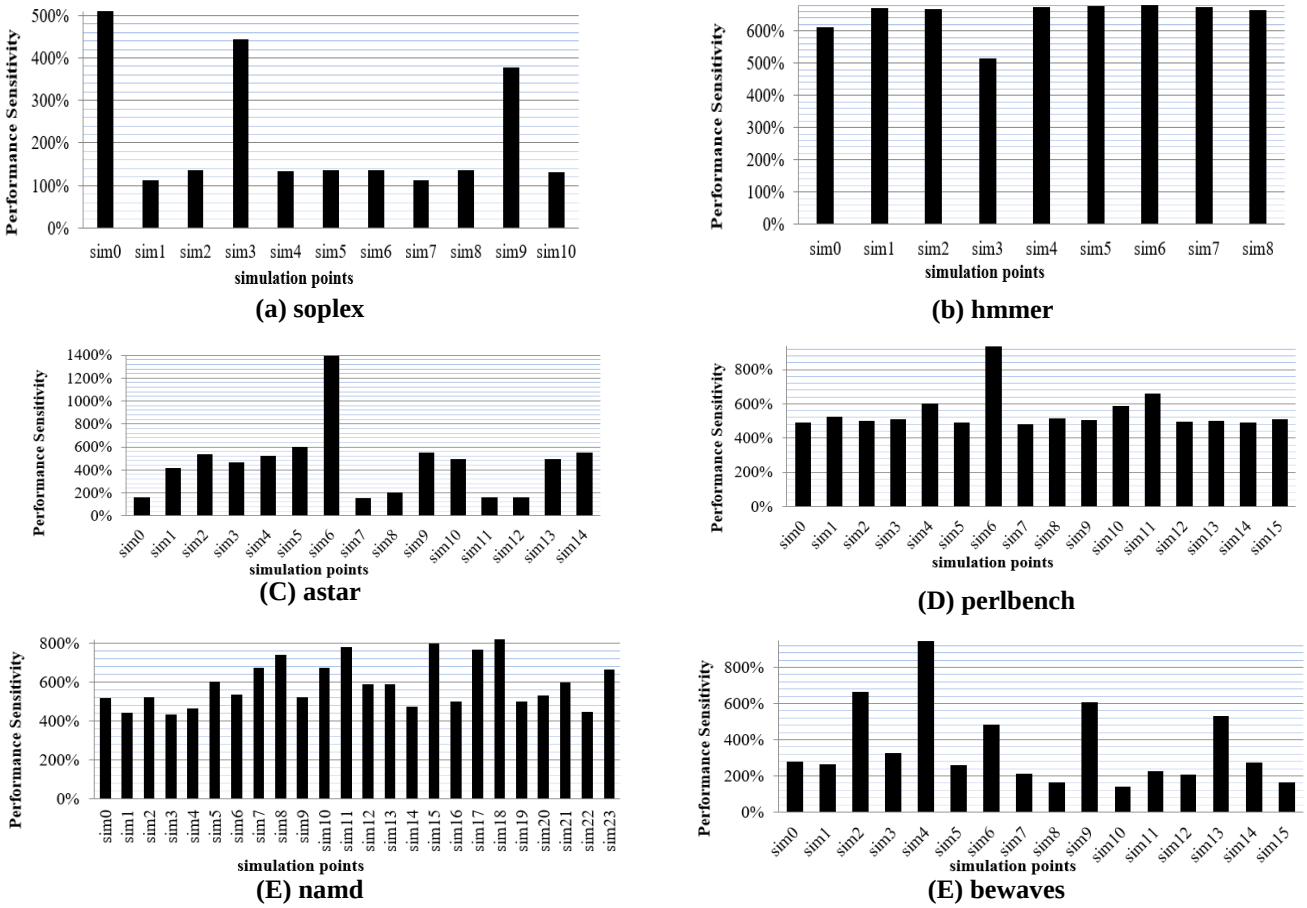


Fig. 2. Degree of performance sensitivity for simulation points

3.3. Workload Pruning with K-mean Clustering

In this research, we choose K-mean technique to cluster simulation points based on their performance sensitivity. K-mean is one of most widely used clustering algorithms, which typically applied to objects in a continuous n-dimensional [25]. We use R language and software environment [26], a professional tool for statistical computing and graphics, to achieve K-mean algorithm. Each simulation points is a 31-dimensinal vector, all simulation points in a certain benchmark can form K clusters/groups by select K points as initial centroids based on their similarity of performance impact. In fact, design exploration should be based on performance variations on different resources or different configurations. If designers need smaller number of simulation points, first thing they need to consider is performance sensitivity to resource variations. The concept, degree of performance-sensitivity, proposed in this paper can be applied. Secondly, they should consider the degree of performance-similarity to reduce the number of simulation points. In this case, weighting factors from the SimPoint needs to be added within the group. One representative simulation point in a group, which has the highest sensitivity within that group, is used to calculate the performance metric with combined weighting values. Table 5-10 show workload pruning based on K-mean clustering.

In the case of astar (Table 7), the first column K represents the selected number of K ($1 \leq K \leq$ No. of simulation points - 1) points, and also equals to the new number of simulations points after pruning. The second and third columns are grouped simulation points and the representative simulation point with highest performance sensitivity in this group. For example, its 15 simulation points can be clustered as 12 groups by selecting K equal to 12. Simulation points (s0, s11 and s12) can be in same PS (performance-sensitivity) group and simulation points (s4 and s9) can be in same PS group. All the other simulation points (s1, s2, s3, s5, s6, s7, s8, s10, s13 and s14) need to be independently evaluated. In the first PS group, the sensitivity of s0 (160.79%) is higher than s11 (157.95%) and s12 (158.44%), so s0 is the representative simulation point in this group. We calculate the percentage of pruned workload over the original workload; the result is workload reduction which means simulation time reduction. The error rate of CPI is reference to the original entire simulation points based on 64 configurations. More workload reduction, higher error rate can be produced. The results from all other benchmarks are shown in Table 5, 6, 8, 9 and 10.

Table 5. Workload Pruning Results (SOPLEX)

K	Grouped Simpoints	Representative Simpoints	Workload Reduction	Error
10	s1, s7	s7	17.18%	0.59%
9	s1, s7	s7	23.59%	0.78%
	s8, s10	s8		
8	s1, s7	s7	29.14%	0.85%
	s2, s4	s2		
	s8, s10	s8		
7	s1, s7	s7	49.71%	0.84%
	s2, s4, s5	s2		
	s8, s10	s8		

Table 6. Workload Pruning Results (HMMER)

K	Grouped Simpoints	Representative Simpoints	Workload Reduction	Error
8	s4, s7	s7	12.64%	0.04%
7	s1, s5	s5	22.95%	0.11%
	s4, s7	s7		
6	s1, s5	s5	31.53%	0.17%
	s4, s7	s7		
	s6, s8	s6		
5	s1, s4, s5, s7	s5	52.48%	0.21%
	s6, s8	s6		

Table 7. Workload Pruning Results (ASTAR)

K	Grouped Simpoints	Representative Simpoints	Workload Reduction	Error
14	s0, s12	s0	9.81%	0.11%
13	s0, s11, s12	s0	19.24%	0.16%
12	s0, s11, s12	s0	24.33%	1.33%
	s4, s9	s9		
11	s0, s11, s12	s0	26.81%	1.26%
	s2, s14	s14		
	s4, s9	s9		
10	s0, s7, s11, 12	s0	29.78%	1.57%
	s2, s14	s14		
	s4, s9	s9		
9	s0, s7, s11,s12	s0	36.97%	2.95%
	s2, s14	s14		
	s4, s5, s9	s5		
8	s0, s7, s11, 12	s0	38.98%	3.01%
	s2, s14	s14		
	s3, s8	s3		
	s4, s5, s9	s5		
7	s0, s7, s11, 12	s0	46.33%	3.49%
	s2, s14	s14		
	s3, s8	s3		
	s4, s5, s9, s13	s5		
6	s0, s7, s11, 12	s0	48.23%	3.48%
	s1, s10	s10		
	s2, s14	s14		
	s3, s8	s3		
	s4, s5, s9, s13	s5		

Table 8. Workload Pruning Results (PERLBENCH)

K	Grouped Simpoints	Representative Simpoints	Workload Reduction	Error
15	s1, s9	s1	3.73%	0.06%
14	s1, s9	s1	8.74%	0.33%
	s2, s13	s2		
13	s1, s9	s1	17.08%	1.43%
	s2, s3, s13	s3		
12	s1, s9	s1	26.66%	1.67%
	s2, s3, s13	s3		
	s5, s14	s14		
11	s1, s9	s1	30.46%	1.74%
	s2, s3, s13	s3		
	s5, s14	s14		
	s10, s11	s11		
10	s0, s12	s12	38.83%	1.65%
	s1, s9	s1		
	s2, s3, s13	s3		
	s5, s14	s14		
	s10, s11	s11		
9	s0, s12	s12	47.85%	3.82%
	s1, s9	s1		
	s2, s3, s5,s13, s14	s3		
	s10, s11	s11		
8	s0, s8, s12	s12	54.66%	3.55%
	s1, s9	s1		
	s2, s3, s5,s13, s14	s3		
	s10, s11	s11		

Table 9. Workload Pruning Results (NAMD)

K	Grouped Simpoints	Representative Simpoints	Workload Reduction	Error
23	s0, s5	s5	1.99%	0.08%
22	s0, s5	s5	5.46%	0.17%
	s6, s19	s6		
21	s0, s5	s5	14.11%	0.19%
	s3, s22	s22		
	s6, s19	s6		
20	s0, s5	s5	23.00%	0.25%
	s2, s13	s13		
	s3, s22	s22		
	s6, s19	s6		

19	s0, s5	s5	22.44%	0.33%
	s2, s13	s13		
	s3, s22	s22		
	s6, s19	s6		
	s16, s20	s20		
18	s0, s5	s5	33.20%	0.71%
	s2, s13	s13		
	s3, s22,s16,s20	s20		
	s6, s19	s6		
17	s0, s5	s5	34.73%	0.65%
	s1, s4	s4		
	s2, s13	s13		
	s3, s22	s22		
	s6, s19	s6		
	s7, s11	s11		
	s16, s20	s20		
16	s0, s5	s5	44.93%	1.03%
	s1, s4	s4		
	s2, s13	s13		
	s3, s22,s16,s20	s20		
	s6, s19	s6		
	s7, s11	s11		
15	s0, s5	s5	47.91%	1.53%
	s1, s9	s9		
	s2, s13	s13		
	s3, s22,s16,s20	s20		
	s6, s19	s6		
	s7, s11	s11		
	s10, s17	sin17		
14	s0, s5	s5	55.98%	1.51%
	s1, s4, s9	s9		
	s2, s13	s13		
	s3, s22, s14	s14		
	s6, s19	s6		
	s7, s11	s11		
	s10, s17	s17		
	s16, s20	s20		

Table 10. Workload Pruning Results (BEWAVES)

K	Grouped Simpoints	Representative Simpoints	Workload Reduction	Error
15	s0, s3	s3	3.32%	0.13%
14	s0, s3	s3	11.75%	0.69%
	s11, s14	s14		
13	s0, s1, s3	s3	15.19%	0.88%
	s11, s14	s14		
12	s0, s1, s3, s5	s3	21.95%	1.08%
	s11, s14	s14		
11	s0, s1, s3,s5	s3	30.63%	1.09%
	s8, s15	s8		
	s11, s14	s14		
10	s0, s1, s3,s5	s3	32.06%	1.44%
	s2, s13	s2		
	s8, s15	s8		
	s11, s14	s14		
9	s0, s1, s3,s5, s14	s3	36.01%	1.09%
	s2, s13	s2		
	s8, s15	s8		
	s11, s12	s11		
8	s0, s1, s3,s5, s14	s3	42.75%	1.44%
	s2, s13	s2		
	s8, s10, s15	s8		
	s11, s12	s11		
7	s0, s1, s3,s5, s14	s3	50.09%	1.57%
	s2, s13	s2		
	s7, s11, s12	s11		
	s8, s10, s15	s8		

3.4. Validity Check of the PS-based Workload Pruning

In order to check the validity of performance-sensitivity based workload pruning, we compared CPI values between the proposed scheme and Simpoint methods. Figure 3 shows the analysis results for soplex and hmmer. One representative simulation point in a group, which has the highest sensitivity within a group, is used to calculate the performance metric with combined weighting values. The soplex shows 5.15% error rate on average with 36% pruned workload compared to all simulation points with weight, while hmmer shows 0.22% error rate on average with 44% pruned workload.

IV. RELATED WORKS

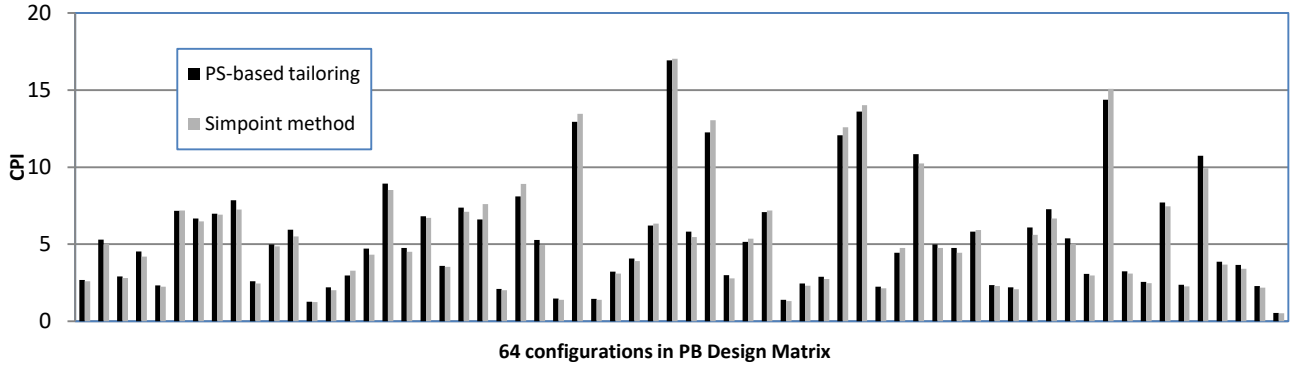
There are several researches that are related to our studies, but we could not find any of that focused on the simulation points classifying for individual benchmark. Most of researches focused on finding the representative of simulation points or classifying of benchmarks. This paper is built upon previous work by using PB design method on simulation points, analyzing and classifying based on performance sensitivity.

Plackett and Burman [2] invented the PB design method to measure the effect of component, it provides the logically minimal number of simulations required to estimate the impact of each selected component with accuracy guarantee.

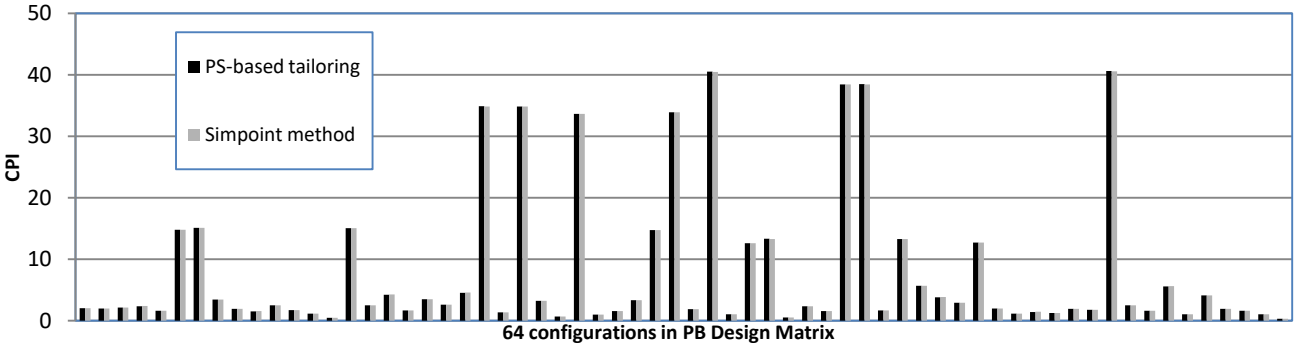
Yi et al. [1] described a method for analyzing and classifying benchmarks which based on the PB design method. Their result shows PB design can significantly reduce the number of simulation. Then, they divide the benchmarks from SPEC2000 benchmarks suit into several groups via the similarity threshold defined by user.

There have also been extensive works to reduce the simulation time in microprocessor design [12][13][14][15][22]. KleinOsowski et al. [13] proposed a method to reduce the simulation time of the SPEC CPU 2000 benchmark suite by using the reduced input data sets. They propose to use small input data sets called MinneSPEC that reflect the behavior of the full input data sets instead of using the reference input data sets provided by SPEC. Eeckhout et al.

12][18] present their analysis results on the impact of input data sets on program behavior using PCA (principal components analysis) and cluster analysis. Phansalkat et al. [8] studied the redundancy of SPEC CPU 2006 benchmark suite based on principal component analysis. Main idea is that SPEC CPU 2006 is biased to some of the applications and simulation time can be reduced by taking benchmarks that are specific to an application. Wunderlich et al. [16] explains about SMARTS, a trace sampling technique for reducing runtimes in simulators but executions still need to handle tens of millions of instructions. Simpoint [7][10] tools are proposed to cluster the simulation points using phase information in dynamic execution, and the PIN tool [9][17][21] is also used for solving the problem of long simulation time.



(a) soplex (error rate: 5.15% on average with 36% pruned workload)



(b) hmmer (error rate: 0.22% on average with 44% pruned workload)

Fig.3.Validity Check of PS-based workload Pruning

V. CONCLUSION

Basic motivation of this research is started from how design exploration is actually performed. Designers will observe the performance impact from resource variations or configuration changes. If a simulation point shows less sensitive to resource variations, designers would eliminate those simulation points from the simulation setup procedure. In this paper, we focus on identifying those simulation points which are more sensitive or less sensitive, by which overall simulation methodology can be improved. We also performed the performance-sensitivity-based similarity analysis (grouping) among simulation points on specific performance metric which can be overall performance metric or component-level metric. The proposed pruning methods are checked its validity with the analysis on the degree of sensitivity. The error rate on average is 0.22%~5.15% with 36%~44% pruned workload compared to all simulation points with weight. Our experiments also show that the proposed methodologies can be applied to simulation points of multiple applications for multi-threaded processors or multi-core environments.

As future works, the proposed schemes will be applied to simulation points of multiple applications for multi-threaded or multi-core simulation workload pruning.

REFERENCES

1. Joshua J. Yi, David J. Lilja, Douglas M. Hawkins, A statistically rigorous approach for improving simulation methodology, *International Symposium on High-Performance Computer Architecture (HPCA)*, February, 2003
2. R. Plackett and J. Burman, “The Design of Optimum Multifactorial Experiments", Biometrika, Vol. 33, Issue 4, June 1956, Pages 305-325
3. D. Lilja, “Measuring Computer Performance”, Cambridge University Press, 2000
4. J. Yi and D. Lilja, "Effects of Processor Parameter Selection on Simulation Results", MSI Report 2002/146, 2002
5. D. C. Montgomery, “Design and Analysis of Experiments”, Third Edition, Wiley 1991
6. Standard Performance Evaluation Corporation (SPEC) website,
7. http://www.spec.org/
8. A. Nair and L. John, “Simulation Points for SPEC 2006,” International Conference on Computer Design (ICCD'08). October 2008
9. A. Phansalkar, A. Joshi and L. K. John, “Analysis of Redundency and Application Balance in the SPEC CPU2006 Benchmark Suite,” The 34th International Symposium on Computer Architecture (ISCA). June 2007
10. PIN home page: http://rogue.colorado.edu/Pin/
11. G. Hamerly, E. Perelman, J. Lau, and B. Calder, “SimPoint 3.0: Faster and More Flexible Program Analysis ,” Workshop on Modeling, Benchmarking and Simulation, June 2005

12. T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. “Automatically Characterizing Large Scale Program Behavior,” Proc. International Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 45–57, Oct. 2002
13. http://cseweb.ucsd.edu/~calder/simpoint/single-sim-pionts.htm
14. L. Eeckhout, R. H. Bell, B. Stougie, K. Bosschere and L. K. John, “Control Flow Modeling in Statistical Simulation for Accurate and Efficient Processor Design Studies,” ISCA. pp. 350-361 2004
15. A. J. KleinOsowski and D. J. Lilja, “MinneSPEC: A New SPEC Benchmark Workload for Simulation-Based Computer Architecture Research,” Computer Architecture Letters, vol.1, May, 2002
16. K. Lee, S. Evans, and S. Cho "Accurately Approximating Superscalar Processor Performance from Traces,” Proceedings of the ISPASS, pp. 238~248, Boston, Massachusetts, April, 2009
17. K. Ganesan, J. Jo, and L. K. John, “Synthesizing Memory-Level Parallelism Aware Miniature Clones for SPEC CPU2006 and ImplantBench Workloads,” ISPASS, March, 2010
18. R. E. Wunderlich, T. F. Wenisch, B. Falsafi, J. C. Hoe, “SMARTS: accelerating microarchitecture simulation via rigorous statistical sampling,” Proceedings. 30th Annual International Symposium on Computer Architecture, pp. 84-95 ,June, 2003
19. H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun and A. Karunanidhi, “Pinpointing Representative Portions of Large Intel Itanium Programs with Dynamic Instrumentation,” In Proceedings of the 37th Annual IEEE/ACM international Symposium on Microarchitecture, 2004
20. L. Eeckhout, H. Vandierendonck and K. Bosschere, “Quantifying the Impact of Input Data Sets on Program Behavior and its Applications,” Journal of Instruction-Level Parallelism, vol. 5, pp. 1-33, 2003
21. D. C. Burger and Todd M. Austin, “The Simplescalar Tool Set, Version 2.0,” UW Madison Computer Sciences Technical Report #1342, 1997
22. D. B. Noonburg and J. P. Shen. “A Framework for Statistical Modeling of Superscalar Processor Performance,” Proc. Int’l Symp. High-Performance Computer Architecture (HPCA), pp. 298–309, Feb.1997
23. C. Luk, R. zohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. Reddi, and K. Hazelwood, “Pin: building customized program analysis tools with dynamic instrumentation,” In Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI ’05. ACM, pp. 190-200, 2005
24. R. H. Bell and L. K. John, “Improved Automatic Testcase Synthesis for Performance Model Validation,“ 19th ACM International Conference on Supercomputing, June 2005
25. S. Raghunath and B. Lee, “Selection of Representative Simulation Point using Performance Metric based Similarity,” Sixth workshop on Unique Chips and System (UCAS-6), December 2010
26. L. Eeckhout, H. Vandierendonck,and K. De Bosschere, “Workload Design: Selecting Representative Program-Input Pairs,”InternationalConference on Parallel Architectures and Compilations Techniques, 2002
27. P. Tan, M. Steinbach, and V. Kumar, “Intruction to Dada Mining”, pp.496-513, 2006
28. R language and enviroment website, http://www.r-project.org/

Byeong Kil Lee, received the Ph.D. degree in computer engineering from the University of Texas at Austin, Austin, in 2005. He is currently an assistant professor in the Electrical and Computer Engineering Department, University of Colorado, Colorado Springs. His current research interests include computer architecture, application-specific embedded systems, low power mobile processors, workload characterization of emerging applications, machine learning and cyber security.