On Performance Binning of Multi-core Processors

John Sartori†, Ashish Pant‡, Rakesh Kumar†, Puneet Gupta‡

†Coordinated Science Laboratory
1308 West Main St
Urbana, IL 61801

‡Electrical Engineering Department
University of California, Los Angeles
Los Angeles, CA 90095-1594

***Abstract*— Number of cores per multi-core processor die, as well as variation between the maximum operating frequency of individual cores, is rapidly increasing. This makes performance binning of multi-core processors a non-trivial task. In this paper, we study multi-core binning metrics and strategies to evaluate them efficiently. We also show that variation model aware binning can significantly reduce the binning overhead with a negligible loss in binning quality.**

I. Introduction

Performance (or speed) binning refers to test procedures to find out the maximum operating frequency of a processor. It is common practice to speed bin processors for graded pricing. In the case of uniprocessors, performance of a processor strongly correlated with its frequency of operation. For multi-core processors, however, the appropriate binning metrics are much less clear.

- Because of large number of cores per die and increasing process variations [5][1], it is unlikely that the maximum operating frequencies of all cores would be similar. Therefore, if binning is done according to lowest common operating frequency of all cores (one obvious extension to the uniprocessor binning metric), it would be highly pessimistic.
- Time overhead for binning is an important metric and a choice needs to be made between the correlation to performance and binning overhead for multi-core processors.

In this paper, we discuss two binning metrics and quantify their correlation with absolute performance and time overhead. We also show that the process variation model itself can be used to come up with binning strategies that take much less time for binning without significantly sacrificing correlation to performance.

II. Modeling Variation

An accurate physically justifiable model of spatial variability is very important to reliably predict and leverage core-to-core variation in the binning process. We use a polynomial variation model similar to ones proposed in [2], [3], [4] having three components: (1) systematic (bowl-shaped) across wafer variation[1]; (2) random core-to-core variation (arising from random within-die variation); and (3) random die-to-die variation (e.g., from wafer-to-wafer or lot-to-lot variation).

$$V_d(x,y) = A(X_c+x)^2 + B(Y_c+y)^2 + C(X_c+x) + D(Y_c+y) + E(X_c+x)(Y_c+y) + F + R + M \quad (1)$$

where $V_d(x,y)$ is the variation of parameter d at die location x,y; X_c, Y_c are the wafer coordinates of the center of the die ($(0,0)$ is center of wafer); x,y are die coordinates of a point within the die; M is the die-to-die variation and R is the random core-to-core variation. A,B,C,D,E,F are fitted coefficients for systematic across-wafer variation. **We use a fitted model as above based on real silicon data from a 45nm industrial process**.[2] The goal of the binning process is to bin a chip into one of n bins (where n is decided based on business reasons) in light of the above variation model.

[1] An example physical source of across-wafer bowl-shaped variation can be plasma etch.

III. Binning Metrics

In this section, we discuss two simple binning metrics that recognize the frequency effects of process variation.

A. Min-Max and Σf

Min-max stands for the minimum of the maximum frequencies for various cores of a chip multiprocessor. The min-max metric is computed with equation 2, where n represents the number of frequency bins, m represents the number of processor cores, and f_{ij} is a successful test frequency or 0 if core j fails the i^{th} test.

$$minmax = \min[\max[f_{ij}|_{i=1}^{n}|_{j=1}^{m} \quad (2)$$

The second binning metric that we look at is Σf which ranks processors based on maximum attainable throughput [6] (every core operating at its maximum frequency).

$$\Sigma f = \sum_{j=1}^{m} \max[f_{ij}|_{i=1}^{n} \quad (3)$$

In terms of correlation of the metric with the throughput of the chip, min-max is a conservative metric and, therefore, should demonstrate good correlation only for workloads with regular partitioning in which the load is distributed evenly between all cores. For other workloads that have inherent heterogeneity or that are multiprogrammed, Σf should demonstrate good correlation.

To compare the two metrics, consider the asymptotic case of very large n and m and completely random core-to-core variation (i.e., A, B, C, D,E, F, M all zero in equation 1). In this simplified case, Σf converges to $m \times mean\_frequency$ while $minmax$ converges to $(E(Min_{i=1\ldots\infty} f_i) = 0$, i.e., we expect the minmax to be a progressively worse metric as number of cores in a die increases or the variation increases.

B. Binning Overhead

To calculate binning overhead for min-max for n frequency bins and m cores, we use binary search[3] to find out f_{max} for every core. However, the search range will reduce progressively. The worst case arises when f_{max} for every core is 1 binsize less than the previously found f_{max} for the last core. In this case, the worst-case tests that need to be performed can be computed as $(log(n!)+m-n)$ (assuming $m \geq n$). The best case binning overhead for min-max would be m. Calculating the average-case binning overhead for min-max for our binning strategy is fairly involved since test time of

[2] For this model mean = 4GHz, $\sigma_{bowl} = 0.128$GHz, $\sigma_R = 0.121$GHz, $\sigma_M = 0.09$GHz.

[3] In this work, we assume that if a core works at a certain frequency, it is guaranteed to work at frequencies below it. This stems from the specific case of using binary search in conjunction with minmax metric. The constraint can be easily avoided by adding one more test per core(i.e., testing it at the minmax frequency

core i depends on minmax frequency seen for the first $i-1$ cores. Assuming n to be large enough for us to approximate (testable) frequency distribution as continuous, the average test time can be derived as

$$T = E(\Sigma_{i=1\ldots m} log(min(x_1 \ldots x_i))$$
$$= E(\Sigma_{i=1\ldots m} log(i \times f(x) \times (1-F(x))^{i-1})$$

where $f(x)$ and $F(x)$ are PDF and CDF of the core frequency distribution respectively. To best of our knowledge no closed form expressions are known for the above. We therefore show the average case testing time by our experiments rather than analytical expressions.

To fully evaluate the Σf metric, the maximum operating frequency of each core must be learned. Using a binary search, this process performs, on the worst, $m \times logn$ tests. The best case is still m tests. The average case binning overhead is $m \times (logn - 1)$.[4]

IV. Using Variation Model to Reduce Binning Overhead

In this section, we argue that the overhead of binning can be considerably reduced by making the binning strategies variation model-aware. f_{max} of a core can be strongly predicted (i.e. mean with standard deviation around it) based on the process variation model. Therefore, process variation model can give a smaller frequency range within which the search should be done.

A. Curve Fitting

The curve-fitting strategy involves approximating the expected frequency (in GHz) as well as the standard deviation ($= \sqrt{(\sigma_M^2 + \sigma_R^2)}$) of a core given its location within a die and die location within the wafer using the variation model (equation 1). Therefore, we can identify center (= mean) as well as corners (= +/-$k\sigma$) of the search range. If the core falls out of this range (decided by k), we assume lowest frequency bin for the core. This reduces both the average and worst-case test time for the core.

B. Clustering

Another strategy for reducing the binning overhead can be clustering the cores in a chip multiprocessor and then using min-max within the cluster (low binning overhead advantage) while using Σf among the clusters (high correlation to maximum throughput advantage). To further reduce the overhead of binning, a process like curve fitting can be applied where the process variation model is used to identify the search range for f_{max} of a core.

In order to improve the performance correlation within the cluster (especially when across-wafer variations are high), clusters can also be chosen intelligently to minimize frequency variation (and hence loss of correlation) within a cluster. To this end, the cluster size can be set to be inversely proportional to the spread of frequency mean (calculated from the bowl-shape in equation 1) within the cluster. In general, the dies close to the center of the bowl (typically close to the center of wafer) will see clusters of large sizes, while clusters are smaller for the dies closer to the edge of the wafer. We do not evaluate variable clustering in this paper due to the relatively low across-wafer variations that our current process variation models suggest.

[4]Note that this expression and the expressions for minmax ignore the bias introduced in binary search by the probability distribution of the frequencies themselves.

TABLE I
Benchmarks used

Program	Description
ammp	Computational Chemistry (SPEC)
crafty	Game Playing:Chess (SPEC)
eon	Computer Visualization (SPEC)
mcf	Combinatorial Optimization (SPEC)
twolf	Place and Route Simulator (SPEC)
mgrid	Multi-grid Solver: 3D Potential Field (SPEC)
mesa	3-D Graphics Library (SPEC)
groff	Typesetting package (IBS)
deltablue	Constraint Hierarchy Solver (OOCSB)
adpcmc	Adaptive Differential PCM (MediaBench)

V. Methodology

We model chip multiprocessors with various number of cores on the die for different technologies. Each core is a dual issue Alpha 21064-like inorder core with 16KB, 2-way set-associative Icache and DCache. Each core (1 mm^2 at 45nm) on a multiprocessor has a private 1MB L2 cache ($0.33MB/mm^2$ at 45nm). We assumed a gshare branch-predictor with 8k entries for all the cores. The various miss penalties and L2 cache access latencies for the simulated cores were determined using CACTI [7]. We model the area consumption of the processors for different technologies using the methodology in [8].

We considered two types of workloads - multiprogrammed workloads and parallel workloads. Table I lists the ten benchmarks used for constructing multiprogrammed workloads. The benchmarks are chosen from different suites (SPEC, IBS, OOCSB, and Mediabench) for diversity. Three parallel applications (CG, FT, MG) are chosen from the NAS benchmark suite and run to completion. The class B implementations are run.

Multiprogrammed workloads are created using the sliding window methodology in [10]. For multiprogrammed workloads, the performance of a multiprocessor is assumed to be the sum of the performance of each core of the multiprocessor derated by a constant factor. The methodology is accurate for our case where each core is assumed to have a private L2 cache and a memory controller [8]. The methodology was shown to be reasonable for our benchmarks even for processors with shared L2 [8] due to the derating factor. Parallel applications as mentioned before are run to completion.

Simulations are done for 250 million cycles for cores running at different frequencies given by the variation model, after fast-forwarding an appropriate number of instructions [9]. Simulations use a modified version of SMTSIM [10].

VI. Analysis of Results

We run Monte-Carlo simulations using 100,000 dies, each die being a 64 core processor (256 mm^2) in a 45nm technology 300mm wafer, binned using 8 frequency bins.

Figure 1 shows the binning overhead and throughput correlation for different number of frequency bins for multiprogrammed workloads. Correlation is calculated using 100,000 data points (processor dies) between the average of the maximum throughput of the various multiprogrammed workloads on a processor (where cores run at different frequencies dictated by the variation model) and the value of the metric when following a given binning strategy. Note that performance of a thread often does not vary linearly with frequency due to pipeline hazards, memory accesses, etc., so correlation will unlikely be 1 for any binning metric.

There are several things to note in the graph. First, Σf has a significantly better correlation to throughput than *minmax* for multiprogrammed workloads. This is not surprising considering that throughput of a thread often depends on the frequency of a core it

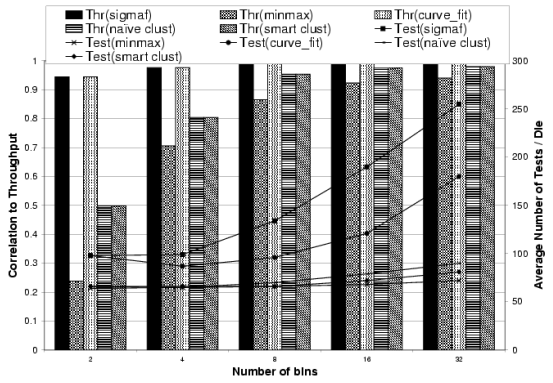

Fig. 1. Effect of maximum number of frequency values a core is tested at, on binning.

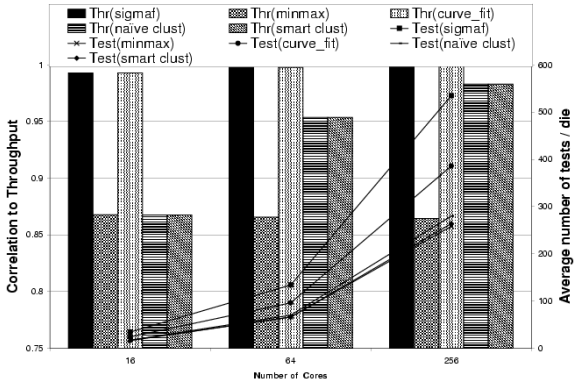

Fig. 2. Effect of varying number of cores per die on binning.

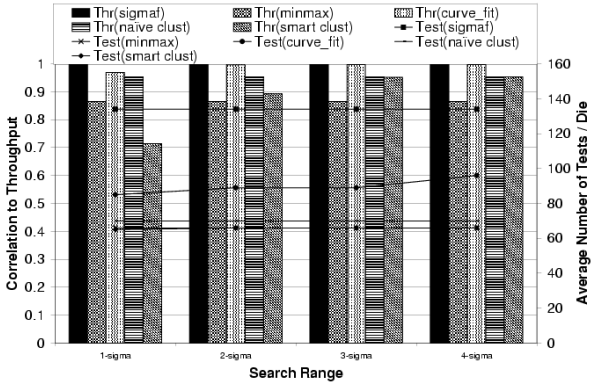

Fig. 3. Effect of binary search range on binning strategies. Search range is varied from $\pm 1\sigma$ to $\pm 4\sigma$ (baseline). Here σ refers to total standard deviation of die-to-die and core-to-core variation.

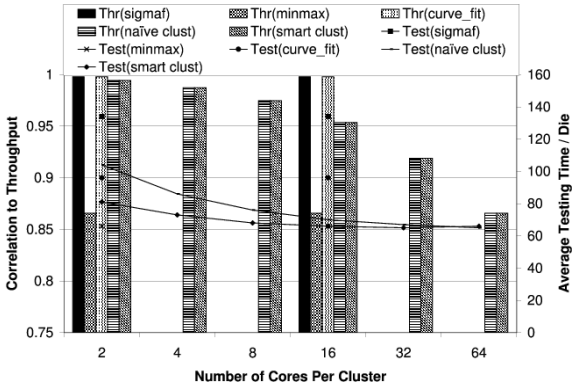

Fig. 4. Effect of varying cluster sizes on binning strategies.

is running on. *minmax* fails to account for the variation in frequency (and, therefore, average throughout) of individual cores. Correlation is especially low for small number of frequency bins. This is because binning process picks an overly conservative frequency as f_{max} for a core in that case. Even the relative performance of *minmax* (as compared to Σf) worsens as the number of frequency bins is decreased. We also evaluated throughput correlation of the two metrics for our parallel workloads. Correlation to throughput improved for *minmax* for such workloads. In fact, it was higher than Σf for large number of frequency bins (16 or higher). This was expected as the throughput of our parallel benchmarks was determined by the slowest thread.

In terms of binning overhead, *minmax* is significantly faster than Σf, especially for large number of bins (70% faster for 32 bins). This is because while Σf involves doing binary search over full range (over all frequency bins) for *every* core, *minmax* has progressively reducing search ranges. *minmax* and Σf have comparable overheads for small number of bins as the search range is reduced.

The graph also shows that *curve_fit* (refers to the approach of using variation model aware curve fitting strategy to approximate Σf) has performance correlation to throughput that is equivalent to that for Σf. This is because a range of six-sigma (+/- 3-sigma) is searched for *curve_fit* which is often big enough to allow the discovery of true f_{max} of a core. In terms of binning overhead, *curve_fit* is significantly faster than Σf (36% for our baseline architecture). This is because the range of frequencies that are searched for *curve_fit* is directed by the variation model and, therefore, small. Overhead is bigger than that for *minmax* because of the need to estimate the f_{max} for every core.

Clustering-based strategies (refers to the approach of using clustering strategy to approximate Σf) (results are shown for a cluster size of 16) result in a smaller binning overhead than *curve_fit* (26% for baseline). Clustering that relies on the variation model to reduce the range of search for f_{max} for the cores (*smart clust*) is faster than than the one that performs search over the full range for all cores (*naive clust*) (6% improvement in test time for baseline case). In terms of correlation to throughput, clustering based strategies lie between Σf and *minmax*. This is not surprising considering that clustering represents a hybrid between the two schemes. The trends as well analysis were same for our parallel workloads.

Figure 2 shows how the binning overhead changes with the number of cores on the processor chips. The numbers are shown for 16 frequency bins. As we can see, the binning overhead increases with increasing number of cores. More interestingly, the correlation to throughput increases for both clustering-based strategies with the number of cores. Better correlation is a result of having a larger number of clusters per chip (note that the more the clusters, the more the number of f_{max} values being summed). To confirm this, we also performed experiments to see how the correlation and binning overhead change when the number of cores per cluster (and, therefore, the number of clusters) is changed for a fixed sized chip (with 64 cores). Figure 4 shows the results. We indeed observe that the binning overhead of clustering decreases with increasing number of cores per cluster. Similarly, the correlation to throughput increases with increasing number of clusters.

We also quantified the effect of changing the range of search.

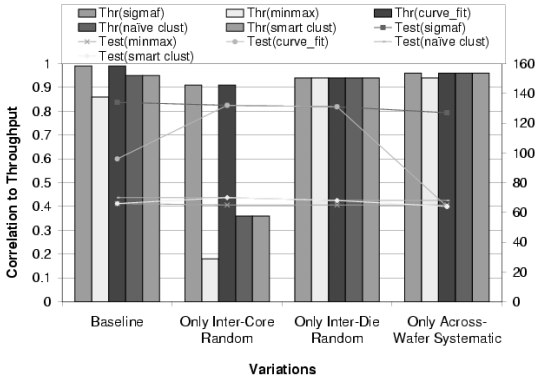

Fig. 5. Effect of nature of variation on binning.

Figure 3 shows how correlation to performance and binning overhead change with the search range. As we can see, both the techniques that rely on the variation model to come up with aggressive search ranges (*curve_fit* and (*smart clust*) have better correlation as the range of search is increased. This is not surprising because the bigger is the range, the higher the probability that a true f_{max} is found.

Finally, we show the effect of nature of variations on binning metrics and their evaluation in Figure 5. The four cases: *baseline* (used in rest of the experiments), *only inter-core random*, *only inter-die random* and *only across-wafer systematic* (i.e., the bowl-shaped variation) all have the same variance. As within-die (i.e. core-to-core) variation increases, correlation of *minmax* to true throughput decreases as it grossly underestimates throughput (since it is taking minimum of f_{max} of all cores). The extremes are *only inter-core variation* where the correlation to throughput is barely 0.18 and *only inter-die variation* where the correlation is same as that of Σf.

VII. Conclusions

In this paper, we have studied for the *first* time, speed binning for multi-core processors. We have compared two intuitive metrics *minmax* and Σf in terms of their correlation to actual throughput as well as testing overhead. Further, we have proposed binning strategies which leverage extent of variation (clustering) as well as partially systematic nature of it (curve fitting). Curve fitting achieves same correlation with true throughput as Σf with significantly less testing overhead (36% for our baseline architecture). Clustering reduces the test time even further (26% for baseline) with little loss in correlation (except for cases when core-to-core random variation is dominant form of variation). We also conclude that the test time benefit of smart clustering is small (6% improvement in test time for baseline case).

Our overall conclusion is that uniprocessor binning methods do not scale well for multi-core processors in presence of variations. Multi-core binning metrics and testing strategies should be carefully chosen to strike a good balance between goodness of the metric and time required to evaluate it. Clustering may be constrained by design decisions (e.g., multiple cores sharing the same clocking) as well as by tester power (in this work we have assumed that only one core can be tested at a time): topics we intend to further pursue.

References

[1] S. Borkar, "Design Challenges of Technology Scaling", *IEEE Micro*, 1999.
[2] K. Qian and C.J. Spanos, "A Comprehensive Model of Process Variability for Statistical Timing Optimization", *Proc. SPIE Design for Manufacturability through Design-Process Integration*, 2008.
[3] P. Friedberg, W. Cheung and C.J. Spanos, "Spatial Modeling of Micron-Scale Gate Length Variation", *Proc. SPIE Data Analysis and Modeling for Process Control*, 2006.
[4] B.E. Stine, D.S. Boning and J.E. Chung, "Analysis and Decomposition of Spatial Variation in Integrated Circuit Processes and Devices", *IEEE Trans. Semiconductor Manufacturing*, 10(1), 1997.
[5] ITRS, 2007, `http://public.itrs.net`.
[6] Asanovic, Krste and Bodik, Ras and Catanzaro, Bryan Christopher and Gebis, Joseph James and Husbands, Parry and Keutzer, Kurt and Patterson, David A. and Plishker, William Lester and Shalf, John and Williams, Samuel Webb and Yelick, Katherine A., "The Landscape of Parallel Computing Research: A View from Berkeley", EECS Department, University of California, Berkeley, 2006.
[7] S. J. E. Wilton and N. P. Jouppi, "CACTI: an enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, 1996.
[8] Rakesh Kumar and Dean M. Tullsen, "Core architecture optimization for heterogeneous chip multiprocessors", *International Conference on Parallel Architectures and Compilation Techniques, PACT*, 2006.
[9] Timothy Sherwood and Erez Perelman and Greg Hamerly and Brad Calder, "Automatically Characterizing Large Scale Program Behavior", ASPLOS, 2002.
[10] D.M. Tullsen, "Simulation and Modeling of a Simultaneous Multithreading Processor", 1996, *22nd Annual Computer Measurement Group Conference*.