Statistical Error Bounds for Data Parallel Applications

Parker Hill   Michael Laurenzano   Babak Zamirai   Mehrzad Samadi
Scott Mahlke   Jason Mars   Lingjia Tang
University of Michigan
{parkerhh, mlaurenz, zamirai, mehrzads, mahlke, profmars, lingjia}@umich.edu

Abstract

Recent approximate computing techniques can trade small amounts of accuracy for very large performance and energy improvements. However, these current techniques are unable to provide error bounds or require careful derivation of error properties for the specific algorithm. For approximate computing to be more widely adopted, increasing the set of approximations where the error can be bounded is critical. In this paper we propose a methodology to statistically bound the error of data parallel computations for approximate computing. We are able to produce very accurate error bounds for the binarize benchmark while computing only a small fraction of the output and without incorporating any application-specific knowledge into the accuracy computation.

1. Introduction

Substantial performance and energy improvements can be achieved by current approximate computing techniques. To increase the adoption of these techniques, some recent work has looked at providing mechanisms to make the accuracy degradation from approximation more predictable (Khudia et al. 2015; Ringenburg et al. 2015; Goiri et al. 2015; Sampson et al. 2014; Samadi et al. 2013; Misailovic et al. 2011). These works provide a more structured approach to reasoning about accuracy, which improves the usability of the underlying approximate computing techniques. Despite these works, the accuracy properties of many computational models and patterns remain unexplored.

In this work we describe a data parallel model and propose a methodology to place statistical error bounds on approximations applied to it. There are a wide range of applications that use this data parallel model because it only requires that outputs can be efficiently computed independently of each other. This independence allows us to reason about accuracy with finer granularity, since each independent output corresponds to an independent error between the exact and approximate computation. The final error can be computed by some aggregation of the error components. This composition of errors into a final error provides us the ability to apply statistical tools to the error components to accurately model the final error. With the final error likelihood function, we can produce a statistical bound on the error.

We show how to derive the error likelihood function and statistical error bound for one error metric, miss rate, which denotes the percentage of values that are incorrect when approximated. We evaluate the error bound that is produced with only 1% of the total computation and find a very close match to the analytical results.

2. Computational Model

We use a data parallel computational model which is found in many domains, including image processing, computer vision, scientific computing, and machine learning. This model requires the outputs of the application to be produced independently of the others and in any order. Based on this data parallelism, we define the terms output components and error components. An output component is the smallest amount of work that the application can execute based on the granularity of its data parallelism. The error component, e_i, is the error found by comparing the output component of the exact computation, o_i, and the approximate computation, a_i.

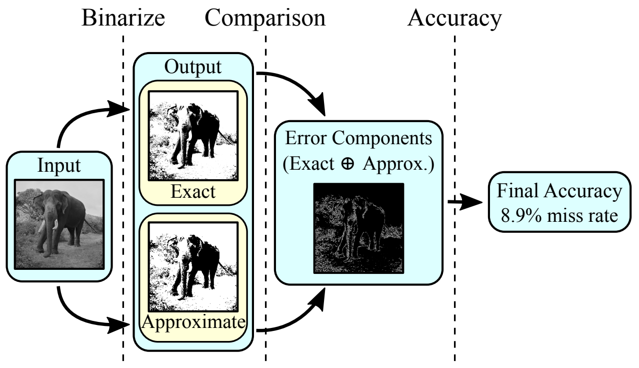


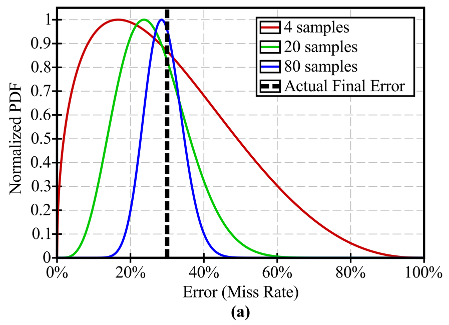
Figure 1: The accuracy computation of an approximation applied to a data parallel application: outputs are computed for exact and approximate algorithms, the element-wise error is computed, and then error components are aggregated to find the final accuracy.

We show a data parallel image processing algorithm, binarize, in Figure 1. In this case, each output pixel of the filter is an output component. The error components can be found by comparing the output of the exact and approximate version of the filter. We define the error component metric to be a binary comparison, where it is 0 when the exact output component matches the approximate output component and 1 otherwise, $e_i = |sgn(o_i - a_i)|$. To compute the final accuracy, the error components are aggregated. For this example, we use the miss rate metric, which represents the percent of incorrect output components (i.e. $\sum e_i/N$ for N components).

3. Error Distribution

Using this computational model, the error components can be randomly sampled, since the output components can be randomly sampled using both the exact and approximate algorithms. These random samples from the error component space can be used to produce an error component distribution. From the error component distribution, we can derive a statistical model for the final error likelihood function and apply existing statistical techniques to it.

Continuing with the binarize example, we present the derivation of the final error likelihood function for the miss rate accuracy metric. By the definition of the binary comparison error component metric, $e_i \in \{0, 1\}$. Therefore, randomly sampled error components follow a Bernoulli distribution with success probability parameter p, the chance of the exact output mismatching the approximate output. Now that the error component can be statistically modeled, we can build upon it to derive a model for the



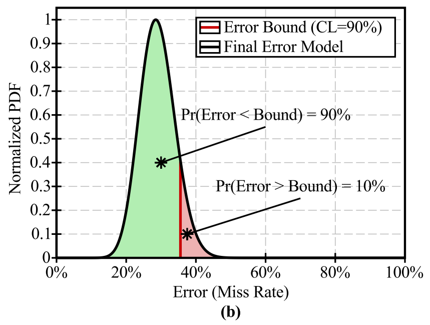


Figure 2: The error bounding process: (a) sampling error components to refine the statistical model representing the final error and (b) the error bound defined by the CL% quantile of the error model.

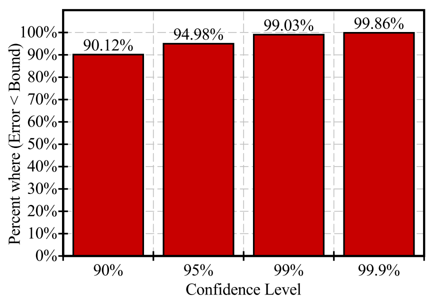


Figure 3: The number of experiments where the error is less than the error bound, showing high quality error bounds from the model.

final error. Since the sum of Bernoulli random variables is equivalent to a binomial distribution with parameters (n, p), where n is the number of error components being summed, we know that $\sum e_i \sim Bin(n, p)$. We note that p is equivalent to our final accuracy metric, miss rate, when the entire set of error components are summed (i.e. $n = N$). We can estimate p by computing a partial sum of error components, $\sum e_i/n$, but this does not allow us to reason about how precise this estimate is, so we would like to build a statistical model for likelihood of p being a given value.

To this end, we can reason about the value of p by modeling it as a random variable, $\tilde{p}$, produced from Bayesian inference based on a set of observed error components. Bayesian inference is a statistical technique where we can refine our knowledge about the random variable $\tilde{p}$ with a set of sampled error components, $(e_1, ..., e_n)$. The formal definition central to this methodology is Bayes' theorem, shown in Eqn. 1, where $f_{P|E}$ is the probability density of the distribution of $\tilde{p}$ given our error component samples and $f_{E|P}$ is the likelihood of observing these samples given the value of $\tilde{p}$. f_P and f_E are the distribution densities of $\tilde{p}$ and the error component samples, respectively, given no knowledge about the variable.

$$f_{P|E}(\tilde{p}|e_1, ..., e_n) = \frac{f_{E|P}(e_1, ..., e_n|\tilde{p}) f_P(\tilde{p})}{f_E(e_1, ..., e_n)} \tag{1}$$

$f_{P|E}$ provides us the refined model of p with a set of error component observations. Using the definition of the error components and the final accuracy metric, the distribution density $f_{P|E}$ can be derived. The final expression of this derivation is provided in Eqn. 3, where B is the Beta function.

$$f_{P|E}(\tilde{p}|e_1, ..., e_n) = \frac{\tilde{p}^{1/2+\sum e_i}(1-\tilde{p})^{1/2+n-\sum e_i}}{\mathrm{B}(\sum e_i + \frac{1}{2}, n - \sum e_i + \frac{1}{2})} \tag{2}$$

This probability density function (PDF) is equivalent to that of the Beta distribution with parameters $(\sum e_i + \frac{1}{2}, n - \sum e_i + \frac{1}{2})$. Therefore $\tilde{p}$ is distributed as shown in Eqn. 3.

$$\tilde{p} \sim \mathrm{Beta}(\sum e_i + \frac{1}{2}, n - \sum e_i + \frac{1}{2}) \tag{3}$$

We show the PDF of this distribution in Figure 2(a) when the miss rate, p, is 30%. It shows the relative likelihood of the miss rate being a specific value, given a set of error component samples. As a larger set of error component samples are used in the statistical model, the PDF becomes more precise (the PDF becomes narrower). It also tends to become more accurate (the center of the PDF is closer to the actual value).

4. Statistical Error Bound

With a statistical model for the likelihood of the final accuracy, we can compute a statistical boundary on the upper limit of the error. To place a useful boundary on the error, a confidence level must be defined. This confidence level denotes how frequently the error boundary can be violated. If complete certainty is needed, then only very loose statements can be made about the boundary. For example, when 10% of the error component space is sampled and all of these components are found to be 0, the only guarantee we can make is that the final error is less than or equal to 90%. Although this is the only possible guarantee, we can intuitively claim that the final error is very close to 0% with very high confidence.

Formally, we state that the confidence level, CL, approaches Pr(Final Error $<$ Bound) as the number of error component samples becomes sufficiently large. For example, in situations where Bound $= 10\%$ and CL $= 99\%$, we expect that 99% of the time the final error is less than 10%. The previously derived final error likelihood function can be used to provide an error bound with this confidence level parameter. As shown in Figure 2(b), the error bound should be placed such that CL (90% in the figure) of the PDF has a lower final error than the the bound. This is equivalent to the quantile of the likelihood function and is readily available for a wide range of distributions using common statistical software.

5. Evaluation

We evaluated the quality of our error bound with the binarize benchmark, a wide range of configurations using the tiling approximation (Samadi et al. 2014), and 800 various images. We experimented with four different confidence levels for verification. In all cases, we used only 1% of the error component sample space.

The percentage of images where the error from the approximation exceeds the bound is presented in Figure 3 for each confidence level. In each case, we can see a very close match between the confidence level and the percent of images that violate the error bound. In addition to controlling the percentage of violations, we note that the violating cases tend to be very close to the bound, since the distribution is very sparse beyond the bound (e.g, Figure 2(b)).

6. Conclusion

Widespread adoption of approximate computing is hindered by the untrusted outcome of approximation in terms of accuracy. Recent works address this problem for certain applications or approximations, but the usage of statistically sound approximation must be widened. We provide a methodology to find statistical error bounds for data parallel applications. Data parallel applications are found in a spectrum of domains, including image processing, computer vision, scientific computing, and machine learning.

References

Í. Goiri, R. Bianchini, S. Nagarakatte, and T. D. Nguyen. Approxhadoop: Bringing approximations to mapreduce frameworks. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 383–397. ACM, 2015.

D. S. Khudia, B. Zamirai, M. Samadi, and S. Mahlke. Rumba: an online quality management system for approximate computing. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 554–566. ACM, 2015.

S. Misailovic, D. M. Roy, and M. C. Rinard. Probabilistically accurate program transformations. In *Static Analysis*, pages 316–333. Springer, 2011.

M. Ringenburg, A. Sampson, I. Ackerman, L. Ceze, and D. Grossman. Monitoring and debugging the quality of results in approximate programs. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 399–411. ACM, 2015.

M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke. Sage: Self-tuning approximation for graphics engines. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 13–24. ACM, 2013.

M. Samadi, D. A. Jamshidi, J. Lee, and S. Mahlke. Paraprox: pattern-based approximation for data parallel applications. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014.

A. Sampson, P. Panchekha, T. Mytkowicz, K. S. McKinley, D. Grossman, and L. Ceze. Expressing and verifying probabilistic assertions. In *ACM SIGPLAN Notices*, volume 49, pages 112–122. ACM, 2014.