Statistical Performance Modeling: Case Study of the NPB 2.1 Results

Erich Strohmaier*

Computer Science Department, University of Tennessee, Knoxville, TN 37996

Abstract. With the results of version 2.1 a consistent set of performance measurements of the NAS Parallel Benchmarks (NPB) are available. Unchanged portable MPI code was used for this set of 269 single measurements. In this study we investigate how this amount of information can be condensed. We present a methodology for analyzing performance data not requiring detailed knowledge of the codes. For this we study several different generic timing models and fit the reported data. We show that with a joint timing model for all codes and all systems the data can be fitted reasonable well. The timing model also contains only a minimal set of free parameters. This method is usable in all cases where the analysis of results from complex application code benchmarks is necessary.

1 Introduction

The set of NAS Parallel Benchmarks (NPB) is one of the best accepted benchmarks for parallel processing [1]. End of 1995 a new version of this suite was released which asked for the first time for performance measurements of the unchanged MPI code which is available from the NASA Ames Research Center. In August 1996 a first set of such results was released [2]. It contained an almost complete set of measurements of 4 codes on 4 different systems for 3 different problem sizes. Such a homogeneous big set of performance data from application codes is the optimal starting point for any in depth analysis of the benchmark and systems in the set.

The number of 269 single measurements immediately brings up the question if and how this amount of information can be condensed. In previous studies we already showed that for the fully vendor optimized NPB 1.0 results a limited number of benchmarks would be sufficient and that Amdahl's Law describes these results very well [3].

In this paper we study to which extend this is true for the measurements of unchanged portable MPI code. We propose a generic timing model with a minimal number of free parameters and fit this model to the data using nonlinear regression. We evaluate the quality of our model by comparison with simpler models with more free parameters and careful examination of the statistical properties of the obtained fits[2]. A basic introduction to the statistics terminology used may be found in [4].

* e-mail: erich@cs.utk.edu

[2] All analysis discussed in this paper are done with the SAS statistical software package

2 Timing models for single systems

Starting point for our analysis are the measured execution times $T_{s,c}$ of a sample of n application codes c which are measured on m different systems s. These times are functions of the number of processors p and the problem size $\mathbf{n}$. For simplicity reasons we consider in the following only the dependency on p and not on $\mathbf{n}$.[3]

The measured times can be separated in times for different computational phases of the execution during which basic types of computational work j like parallel computation, serial computation or communication take place. We split each of these basic types of work j into a sum of products. One factor $u_i(p)$ of each term contains all the dependencies on p and is indexed independent of code or system. We normalize these functions such that $u_i(1) = 1$ or $u_i(1) = 0$. The other factors $t_{j,i}^{s,c}$ are parameters depending on code and system.

$$T_{s,c}(p) = \sum_{j=1}^{n} t_j^{s,c}(p) = \sum_{j=1}^{J} \sum_{i=1}^{I} t_{j,i}^{s,c} u_i(p) \tag{1}$$

We call u_i the "characteristic functions" as they contain all the dependencies of the performance on p. They therefor characterize the scaling behavior of the code with increasing processor number p. The set of the characteristic functions u_i reflects the typical algebraic form of timing relations for parallel computing.

We now continue by using the following general timing model for analyzing the measured data:

$$T_{s,c}(p) = \sum_{i=1}^{I} \delta_i^{s,c} u_i(p) \quad \text{with} \quad \delta_i^{s,c} = \sum_{j=1}^{J} t_{j,i}^{s,c} \tag{2}$$

If we analyze the performance data of a single code on a single system we will only be able to fit the sums $\delta_i^{s,c}$. With statistical methods we will not be able to gain information about the individual $t_{j,i}^{s,c}$ [5]. This can be achieved only by analyzing the results of a set of codes measured on a set of systems as done in section 5.

We now want to use equation 2 to analyze the performance results of an application without inspecting the code. For this it is critical to select a reasonable set of characteristic functions u_i such that the set can effectively characterize the execution times of parallel applications. This set of functions together with the number and quality of measurements will determine how many of the parameters $\delta_i^{s,c}$ can be fitted in a meaningful way. As characteristic functions for the further analysis we use the following set which is collected from different sources [6, 7, 5]

$$u_1 = \frac{1}{p^2}, \quad u_2 = \frac{1}{p}, \quad u_3 = \frac{log(p)}{p}, \quad u_4 = \frac{1}{\sqrt{p}}, \quad u_5 = 1, \quad u_6 = log(p), \quad u_7 = p \tag{3}$$

[3] In principal a similar approach as ours can be chosen to account for the dependency on the problem size $\mathbf{n}$.

3 Preparation of the data

Looking on the execution times of the NPB 2.1 we notice a big range in the measured times[4]. For class A problem size measured times range from 0.75 seconds up to 4873.7 seconds. This range of 4 magnitudes of orders poses a severe scale problem for any statistical method and a transformation of this scale is required for statistical reasons. We are using the following transformations.

One major source for the differences in measured times are the different number of processors used. As we are dealing with well parallelized codes multiplication of the measured times with the number of processor[5] has a smoothening effect on the data.

The variation in the floating-point operation count is one reason for the differences between the execution time of the different codes. Scaling measured times with the inverse of this operation count equalizes the scale for the different codes further.

Using processor with different computational power is another reason for differences in execution times. A multiplication of the measured times with peak performance r_{peak} provides additional correction of the scale.

Applying all three transformation in sequence the measured execution times $t(p)$ get transformed in a value $t'(p)$ which is given as

$$t'(p) = \frac{t(p) * p}{W_c} * r_{peak}. \tag{4}$$

t' is a dimensionless value which can be interpreted as inverse temporal efficiency. The values of t' now vary by a moderate factor of 4 to 7 for the three different problem size classes.

After removing two clear outliers from our sample of results we have for problem size class A and B on the average about 6 to 7 observations for each code on each system. This is sufficient for a statistical analysis. There are however almost no measurements for two of the systems in problem class C which is a clear limitation in the usability of this set of results.

A final inspection of the plotted performance data over the number of processors shows the following observations:

- Measurements for the Cray T3D often show unstable performance values over the number of processors. Performance values per processor can drop or rise by about 30% for measurements with similar processor numbers.
- The PowerChallenge Array shows clearly two regimes of operation associated with it's hierarchical architecture. Performance within a single SMP node tend to show super linear speedup while performance between SMP nodes can drop significantly.

These two facts are limits for our analysis as none of our characteristic functions from equations 3 can model such behavior effectively.

[4] see http://www.nas.nasa.gov/NAS/NPB/

[5] For our analysis we are using the number of active processors and not the number of allocated processors as done in the NPB report [2].

u_1	u_2	SSE	R^2	δ_1	δ_2
$log(p)/p$	$1/p^2$	26.13	0.99289	3.616	64.056
$1/p$	$1/\sqrt{p}$	26.40	0.99282	11.595	0.558
$1/p$	1	28.06	0.99237	14.066	0.027
$1/p$	$log(p)/p$	28.17	0.99234	6.443	2.422
$1/p$	$log(p)$	29.18	0.99206	14.516	0.005
$1/p$	p	35.41	0.99037	15.304	0.000

Table 1. All two parameter models with $R^2 \geq 0.99$ for the class A SP results on the Cray T3D. The total Sum of Squares (SST) is 3677.85. SSE: Sum of Squares of the remaining Error; R^2: coefficient of determination; δ_i are the fitted model parameter.

4 Results for individual fits

We start the analysis with two parameter models which are a compromise between the very limitations of one parameter models and the limited number of observations available for each analysis. We fit each possible timing model based on two characteristic functions separately for each code and each system to the data. As the number of observations for many cases is quite low (≤ 6) there are always several models which explain the same fraction R^2 of the total sum of squares in the model. As example we show in table 1 all meaningful combinations with an $R^2 \geq 0.99$ for the class A SP results on the Cray T3D. We have chosen this example as it contributes the most to the total Sum of Squares of this problem size class. For class A the Sum of Squares (SS) of SP on the Cray T3D is 3677.85 which is equivalent to 26.4% of the total Sum of Squares (SST) of this class.

Not only in this example but in most cases a precise selection of a single best model is not possible. There is however a clear trend to models containing $u_2 = \frac{1}{p}$ which is characteristic for parallel work. To find out which characteristic functions might be good candidates for a joint model for all codes and systems we fitted the same model to all combinations of code and systems and calculated the total SSE. In table 2 we show the SSE values for the 5 best models for each class together with the SSE value if taking the best individual model for each pair of code and system. Again most of the models contain $u_2 = \frac{1}{p}$. Models which contain a second function characteristic for limited parallelism $u_4 = \frac{1}{\sqrt{p}}$ or $u_3 = \frac{log(p)}{p}$ or for serial work $u_5 = 1$ tend to fit the data better then models including parallel overhead functions like $u_6 = log(p)$ or $u_7 = p$.

5 Joint timing model for all systems

We now proceed the analysis by making the additional assumption that the $t_{j,i}^{s,c}$ from equation 2 are the quotient of factors which only depend on the code $w_{j,i}^{c}$

u_1	u_2	SSE	R^2	u_1	u_2	SSE	R^2
Class A		SST = 13937.17		Class B		SST = 10769.02	
best	best	39.14	0.99719	best	best	85.17	0.99209
$1/p$	1	47.04	0.99663	$1/p$	$1/\sqrt{p}$	95.44	0.99114
$1/p$	$log(p)$	48.59	0.99651	$1/p$	1	96.95	0.99100
$1/p$	$1/\sqrt{p}$	52.39	0.99624	$1/p$	$log(p)$	99.15	0.99079
$1/p$	$log(p)/p$	80.87	0.99420	$1/p$	$log(p)/p$	101.51	0.99057
$log(p)/p$	$1/p^2$	103.91	0.99254	$log(p)/p$	$1/p^2$	124.63	0.98843

Table 2. The SSE values for the best 5 two parameter models used for all observations compared to the SSE value when using the best individual model for each pair of code and system. The best models for class C are the same as for class B.

(amount of work) or the system $r^s_{j,i}$ (power of the system).

$$t^{s,c}_{j,i} = \frac{w^c_{j,i}}{r^s_{j,i}} \tag{5}$$

The total execution time can now be written as

$$T_{s,c}(p) = \sum_{j=1}^{J}\sum_{i=1}^{I} \frac{w^c_{j,i}}{r^s_{j,i}} u_i(p) = \sum_{i=1}^{I} \delta^{s,c}_i u_i(p) \quad \text{with} \quad \delta^{s,c}_i = \sum_{j=1}^{J} \frac{w^c_{j,i}}{r^s_{j,i}} \tag{6}$$

By using this product representation we have introduced an additional degree of freedom for each characteristic function in equation 6. This follows as each $\delta^{s,c}_i$ is invariant if we multiply the values of $w^c_{j,i}$ and $r^s_{j,i}$ for all j by an arbitrary factor. This degree of freedom has to be fixed by an additional condition on the parameters $w^c_{j,i}$ and $r^s_{j,i}$. We choose for this study to fix one of the system parameters w^c equal to 1. This additional degree of freedom also implies that the absolute values of the parameters $w^c_{j,i}$ and $r^s_{j,i}$ by them self have no meaning as they can be manipulated by changing the normalization. Only the ratios of these parameters are invariant to such changes and can be interpreted in a safe way.

Analyzing the full set of results $T_{s,c}$ we can now fit values to the individual $w^c_{j,i}$ and $r^s_{j,i}$. The two sets of parameters work $w^c_{j,i}$ and speed $r^s_{j,i}$ together with the characteristic functions $u_i(p)$ fully describe the timing models for all codes on all systems included in the analysis.

Overall this product representation reduces the number of free parameters in the analysis effectively by a factor of $\frac{nm}{n+m-1}$ compared to fitting individual models for each pair of the m systems and the n codes. The number of free parameters is indeed quite small as we have for each "type of work" described by the characteristic functions only one parameter for each code and one for each system. This reduction in the free parameters represent the possible value of this model as it potentially can explain the same number of observations with less or even a minimal set of free parameters.

u_1	u_2	SSE	R^2	u_1	u_2	SSE	R^2
Class A		SST = 13937.17		Class B		SST=10769.02	
$1/p$	$1/\sqrt{p}$	130.28	0.99065	$1/p$	$log(p)/p$	134.73	0.98749
$1/p$	1	142.02	0.98981	$1/p$	$1/\sqrt{p}$	156.03	0.98551
$1/p$	$log(p)/p$	146.24	0.98951	$1/p$	1	179.27	0.98335
$1/p$	$log(p)$	148.84	0.98932	$1/p$	$1/p$	180.67	0.98322
$log(p)/p$	$1/p^2$	261.88	0.98121	$1/p$	$log(p)$	185.13	0.98281

Table 3. The 5 best two function models using the product representation from equation 5.

6 Results for the combined model

We now fit all possible timing models of the form of equation 6 based on two characteristic functions to all measurements. It turns out that for problem size class C because of the high number of missing measurements for two of the systems no analysis comparable to class A and B is possible. We compare the results show in table 3 to the results for fitting individual models in table 2.

The values of SSE are higher for the combined model. This was to expected as we now have only 14 free parameters instead of 32. The absolute increase compared to SST is quite small for each problem size class. This is a first strong confirmation that our factorization assumption from equation 5 works quite well.

We discuss now three of the overall best models in more detail. All three models contain $u_2 = \frac{1}{p}$ as first characteristic function. As second function they contain $u_5 = 1$ or $u_4 = \frac{1}{\sqrt{p}}$, $u_3 = \frac{log(p)}{p}$. This sequence of second functions is equivalent from going from serial work to better and better parallel execution.

In table 4 we show the actual fitted values for the 14 free parameters together with their asymptotic standard error for the class A and B. The parameters are fitted for the transformed t' from equation 4 and can be interpreted as the inverse of processor efficiencies and as code overhead factors. The absolute value of the parameters is however without any meaning. Only appropriate chosen ratios of them represent measurable values.

We notice that the standard errors for most parameters are in the range of 5% to 20% of the fitted value. For Class A only the second system parameter of the SGI PowerChallenge Array shows quite big error bars. This is certainly related to the previous mentioned special behavior of the measured data for this system. For class B the same is true for the second system parameter of the Intel Paragon. As an inspection of the measured data shows no special behavior of this system the most likely explanation of this large error is the small number of measurements for this system (16 out of 102).

For all systems the first system parameter varies only little between the three different models. If we interpret it as computational power then the IBM SP2 shows always performance efficiencies twice as large as the other systems. This

Parameter	Class A			Class B		
	$u_1 = 1/p$			$u_1 = 1/p$		
r_1^{IBMSP2}	1.	1.	1.	1.	1.	1.
$r_1^{Paragon}$	0.406 ± 0.026	0.370 ± 0.032	0.365 ± 0.041	0.401 ± 0.033	0.382 ± 0.038	0.381 ± 0.042
$r_1^{CrayT3D}$	0.425 ± 0.024	0.410 ± 0.031	0.466 ± 0.047	0.479 ± 0.035	0.541 ± 0.053	0.669 ± 0.102
$r_1^{PCArray}$	0.482 ± 0.031	0.408 ± 0.037	0.372 ± 0.041	0.457 ± 0.034	0.452 ± 0.041	0.445 ± 0.042
w_1^{BT}	4.572 ± 0.297	4.009 ± 0.365	3.714 ± 0.416	4.840 ± 0.390	4.563 ± 0.426	4.306 ± 0.413
w_1^{LU}	4.023 ± 0.249	3.506 ± 0.297	3.430 ± 0.367	4.227 ± 0.317	4.268 ± 0.386	4.309 ± 0.403
w_1^{SP}	5.355 ± 0.308	4.120 ± 0.337	3.521 ± 0.389	5.729 ± 0.402	5.166 ± 0.460	4.987 ± 0.475
w_1^{MG}	4.672 ± 0.293	3.970 ± 0.347	3.408 ± 0.373	4.049 ± 0.324	3.455 ± 0.375	3.161 ± 0.368
	$u_2 = 1$	$u_2 = 1/\sqrt{p}$	$u_2 = log(p)/p$	$u_2 = 1$	$u_2 = 1/\sqrt{p}$	$u_2 = log(p)/p$
r_2^{IBMSP2}	1.	1.	1.	1.	1.	1.
$r_2^{Paragon}$	1.538 ± 0.360	1.221 ± 0.233	1.012 ± 0.200	6.521 ±18.08	2.902 ± 3.864	1.888 ± 1.829
$r_2^{CrayT3D}$	1.913 ± 0.266	1.106 ± 0.126	0.699 ± 0.079	1.053 ± 0.328	0.537 ± 0.143	0.335 ± 0.088
$r_2^{PCArray}$	2.351 ± 3.628	3.878 ± 4.918	4.037 ± 3.781	0.182 ± 0.086	0.343 ± 0.126	0.422 ± 0.132
w_2^{BT}	0.013 ± 0.006	0.172 ± 0.056	0.480 ± 0.114	0.004 ± 0.005	0.089 ± 0.044	0.271 ± 0.086
w_2^{LU}	0.033 ± 0.007	0.319 ± 0.064	0.577 ± 0.126	0.007 ± 0.006	0.059 ± 0.047	0.158 ± 0.087
w_2^{SP}	0.067 ± 0.007	0.753 ± 0.071	1.535 ± 0.154	0.029 ± 0.009	0.306 ± 0.081	0.615 ± 0.155
w_2^{MG}	0.014 ± 0.007	0.235 ± 0.067	0.750 ± 0.132	0.020 ± 0.007	0.260 ± 0.075	0.548 ± 0.143

Table 4. The fitted parameter with their asymptotic standard error for the best three combined models. The values shown are parameters for transformed t'. This means that system parameters are scaled by the peak performance and code parameters by the single processor floating point operation count.

is not always true in the second set of system parameters.

Looking on the first set of code parameters we see a larger influence of the chosen model on the parameter. This corresponds to the effect of the second functions which represent gradually different limited parallelism. The second code parameter increases as the second function changes to more parallel work. At the same time the first code parameter decreases. If the single processor floating point count which we used for scale transformation would accurately describe the amount of the total computational work then all code parameters should be equal. The values for BT, LU and MG seem indeed to be roughly equal. The values for SP are however consistently higher especially in the second parameter. This indicates that SP contains a substantial additional amount of computational work which is only partially parallelized.

A check of the statistical quality of the obtained fits shows that the correlation matrix of the parameters for the three models have typical values of about 0.5–0.7 for the first parameters and much smaller values for all other entries. The quantile-quantile plots for the errors are quite straight but show typically some outliers at the higher end of the curve. The maximum relative error of the predicted values is about 30% with only one value above 30% and the mean value of the relative error is only 7%.

7 Conclusions

In this paper we present a methodology for analyzing performance measurements without detailed knowledge of the used codes. It is based on the usage of generic timing models build with characteristic function which are typical for the algebraic form of timing equation in parallel computing. We use this methodology to analyze the NPB 2.1 results. Our results can be summarized as follows:

- Using a sequence of transformations solves the statistical scale problems.
- Analyzing each pair of system and code separately between 99.2% and 99.7% of the total Sum of Square (SST) can be explained with individual two parameter functions. This model has 32 free parameters for each class.
- Using a joint timing model with only 14 free parameter 99.1% of the SST of class A and 98.7% of the SST of class B can be explained by this model.
- Typical standard error for the fitted parameter are in the range of 10%. Only one parameter per class is not significantly different from 0.
- The maximum relative error of the predicted values is about 30% and the mean value of the relative error is 7%.
- The average efficiency of the SP2 processor is more than twice as high as for the other processors.
- The simulated CFD application SP contains a substantial amount of work which is not included in the single processor floating point counts.

This methodology for empiric modeling of performance measurements does not require detailed analysis of the implementations of the code. This makes this method to a good alternative in all cases where the analysis of results from complex application code benchmarks is necessary.

References

1. D. Bailey, J. Barton, T. Lasinski, and H. Simon (editors). The NAS parallel benchmarks. Technical Report RNR-91-02, NASA Ames Research Center, January 1991.
2. W. Saphir, A. Woo and M. Yarrow. The NAS parallel benchmarks 2.1 Results. Technical Report NAS-96-01, NASA Ames Research Center, August 1996.
3. Horst D. Simon and Erich Strohmaier. Statistical Analysis of NAS Parallel Benchmarks and LINPACK Results. In Bob Hertzberger and Guiseppe Serazzi, editors, *High-Performance Computing and Networking*, pages 626–633, May 1995.
4. Raj Jain. The Art of Computer Systems Performance Analysis. Wiley, 1991
5. Strohmaier, Erich. Extending the Concept of Computational Similarity for Analyzing Complex Benchmarks. Technical Report 43, Rechenzentrum der Universitaet Mannheim, April ,1995,
6. Vipin Kumar et al.. Introduction to Parallel Computing: Design and analysis of parallel algorithms. Benjamin/Cummings, 1994.
7. Jurgen Brehm and Patrick H. Worley and Manish Madhukar. Performance Modeling for SPMD Message-Passing Programs. Technical Report TM-13254, Oak Ridge National Laboratory, June 1996