Statistical Performance Modeling: Case Study of the NPB 2.1 Results

Erich Strohmaier*

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

UTK-CS-97-354, March 1997

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. This 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 [2]. In August 1996 a first set of such results was released [3]. 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 older fully vendor optimized NPB 1.0 results a limited number of benchmarks would be sufficient [4] and that Amdahl's Law describes these results very well [5].

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

* e-mail: erich@cs.utk.edu, Tel: (+1) (423) 974 0293, Fax: (+1) (423) 974 8296

properties of the obtained fits[2]. A basic introduction to the statistical term used may be found in [6].

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 usually functions of the number of processors p and the problem size characterized by some set of input parameter $\mathbf{n}$. The measured times can be separated in a sum of execution 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.

$$T_{s,c}(\mathbf{n};p) = \sum_{j=1}^{n} t_j^{s,c}(\mathbf{n};p) \tag{1}$$

For simplicity reasons we consider in the following only the dependency on the processor number p and not on the problem size $\mathbf{n}$. In principal a similar approach as ours can be chosen to account for the dependency on the problem size $\mathbf{n}$. In practice this is more difficult as the metric "problem size" is not by itself as well defined as the number of processor. It also requires measurements for a reasonable number of problem sizes which are not available for the NPB.

We now 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 the combination of code and system.

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

We call u_i the "characteristic functions" as they contain all the dependencies of the performance on p and 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. This set of functions is not chosen because of special mathematical properties like orthogonality but because of its relevance for the expected behavior of parallel performance measurements.

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

$$T_{s,c}(p) = \sum_{i=1}^{I} \delta_i^{s,c} u_i(p) \quad \text{with} \quad \delta_i^{s,c} = \sum_{j=1}^{J} t_{j,i}^{s,c} \tag{3}$$

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

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 $t_{j,i}^{s,c}$ [7]. 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 3 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 amount and quality of measured times will determine how many of the parameters $\delta_i^{s,c}$ can be fitted in a meaningful way. Calculating confidence intervals for the fitted parameters and analyzing the total Sum of Squares (SST), the Sum of Squares for the model (SSR) and for the remaining error (SSE) are necessary to judge on the overall quality of the fitted model. We will use the coefficient of determination $R^2 = \frac{SST-SSE}{SST}$ for this purpose.

It is also important to notice that in general not only execution times can be analyzed this way but all performance metrices which can be split up as in equation 3. This especially includes temporal efficiencies and dimensionless temporal efficiencies which can be scaled by additional functions of p. In section 3 we show that for statistical reasons it is preferable to use such derived metrices and not the measured times directly.

As characteristic functions for the further analysis we use the following set which is collected from different sources [8, 9, 10, 5]

$$u_1 = \frac{1}{p^2} \tag{4}$$

$$u_2 = \frac{1}{p} \tag{5}$$

$$u_3 = \frac{log(p)}{p} \tag{6}$$

$$u_4 = \frac{1}{\sqrt{p}} \tag{7}$$

$$u_5 = 1 \tag{8}$$

$$u_6 = log(p) \tag{9}$$

$$u_7 = p \tag{10}$$

3 Preparation of the data

Elapsed execution time is the only system and code independent defined performance metric. Therefor measured execution times are the natural starting point for any performance analysis which attempts to compare different systems with several codes. Looking on the published execution times of the NPB 2.1 we notice a big range in the measured times[3]. For class A problem size (as example) mea-

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

sured times range from 0.75 seconds for MG on a 128 processor IBM SP2 up to 4873.7 seconds for BT on a single processor of the Power Challenge. This range of 4 magnitudes of orders poses a severe scale problem for any statistical method and a transformation of this scale is required. Rather than trying model independent techniques for scale transformation we are using the following sequence of three physically inspired scale transformations.

One major source for the huge differences in measured times is of course the usage of a quite different number of processors between 1 and 256. As we are dealing with well parallelized codes multiplication of the measured times with the number of processor[4] can be expected to have a smoothening effect on the data. The observed range of this total time for class A is between 50 seconds and 30,000 seconds and thus by one magnitude of order reduced.

The floating-point operation count W_c of all the different NPB codes was determined by code inspection and measurement on single processors [3]. The variation in this operation count is the major reason for the differences between the execution time of the different codes. Scaling measured times with the inverse of this operation count tends to equalize the scale for the different codes. This is equivalent to using inverse MFlop/s values if these values are based on a constant operation count. For class A we get a variation form 13 s/GFlop to 250 s/GFlop after this transformation.

Using processor with different computational power is the major reason for the differences between the times on different systems. The only ad hoc available information about the potential performance of a processor is it's peak performance r_{peak}. Even as the peak performance is not a good approximation of the real performance a multiplication of the measured times with peak performance provides some correction of the scale of used values which vary after this transformation between 3.5 and 21 for class A.

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

$$t'(p) = \frac{t(p) * p}{W_c} * r_{peak}. \tag{11}$$

t' is a dimensionless value which can be interpreted as inverse temporal efficiency. It is interesting to notice that Roger Hockney's framework of computational similarity is based on a similar dimensionless temporal efficiency [9]. The values of t' now vary by a factor of 4 to 7 for the three different problem size classes. This value is still moderately high but much smaller than the scale of the original measured times.

Next we have to inspect the sample of t' values for outliers. These are values which represent atypical measurements and are typically created by using a system outside of it's normal mode of operation. While such measurements might still be correct and provide insight in the limitation of a system they can disturb a statistical analysis substantially and therefor have to be removed. In the case

[4] For our analysis we are using the number of active processors and not the number of allocated processors as done in the NPB report [3].

	Class A					Class B					Class C				
	BT	LU	SP	MG	Total	BT	LU	SP	MG	Total	BT	LU	SP	MG	Total
Cray T3D	11	8	12	6	37	9	7	9	5	30	5	3	8	3	19
IBM SP2	11	8	11	8	38	10	8	11	8	37	5	5	10	2	22
Intel Paragon	5	4	7	3	19	3	4	7	2	16	0	1	2	0	3
SGI PC-Array	5	6	5	6	22	4	5	4	6	19	0	3	0	2	5
Total	32	26	35	23	116	26	24	31	21	102	10	12	20	7	49

Table 1. Number of observations in the analysis for each of the three problem size classes.

of the NPB 2.1 measurements there are two clear outliers both for problem size class C on the PowerChallenge Cluster. The measured performances of MG with 32 processor and SP with 16 processor are by one magnitude of order lower than other measurements on this system and we delete these two observations from the subsequent analysis.

The number of observation remaining in the analysis is shown in table 1. For problem size class A and B there are on the average about 6 to 7 observations for each code on each system which 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 4–10 can model such behavior effectively.

4 Results for individual fits

For analyzing measurements for each pair of system and code individually we decided to start with two parameter models. They are a compromise between the very limited usefulness of one parameter models and the limited number of observations available for each analysis. We started fitting each possible timing models based on two characteristic functions separately for each code and each system to the measurement. The set of function we used is given in equations 4–10. As the number of observations for many cases was quite low (≤ 6) there are

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
$1/\sqrt{p}$	$1/p^2$	57.21	0.98444	1.386	170.512

Table 2. All two parameter models with an $R^2 \geq 0.98$ for the class A SP results on the Cray T3D. We show only models which are better than the one parameter models build with one of their characteristic functions u_i. The total Sum of Squares (SST) is 3677.85. SSE means Sum of Squares of the remaining Error, R^2 is the coefficient of determination and a measure for the quality of the used model, δ_i are the fitted parameter.

always several combinations of functions which explain almost the same fraction R^2 of the total sum of squares in the model. As example we show in table 2 all meaningful combinations with an $R^2 \geq 0.98$ for the class A SP results on the Cray T3D. We have chosen this example as this is the pair of code and system which 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 almost all 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 3 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. We again eliminated all models which are worse than one parameter models build with one of their characteristic functions. 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 an additional assumption about the form of the individual times $t^{s,c}_{j,i}$ from equation 3. Based on the physical analogy that time is the quotient of work divided by power we assume that the $t^{s,c}_{j,i}$ are the quotient of factors which only depend on the code $w^c_{j,i}$ (amount of work) or

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

Class C		SST = 4667.79	
best	best	15.06	0.99677
$1/p$	$1/\sqrt{p}$	16.05	0.99656
$1/p$	1	16.16	0.99654
$1/p$	$log(p)$	16.21	0.99653
$1/p$	$log(p)/p$	16.55	0.99646
$log(p)/p$	$1/p^2$	23.64	0.99494

Table 3. 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 system $r^s_{j,i}$ (power of the system).

$$t^{s,c}_{j,i} = \frac{w^c_{j,i}}{r^s_{j,i}} \tag{12}$$

The total execution time is now

$$T_{s,c}(p) = \sum_{j=1}^{J} \sum_{i=1}^{I} \frac{w^c_{j,i}}{r^s_{j,i}} u_i(p) \tag{13}$$

which again can be written as

$$T_{s,c}(p) = \sum_{i=1}^{I} \delta^{s,c}_i u_i(p) \quad \text{with} \quad \delta^{s,c}_i = \sum_{j=1}^{J} \frac{w^c_{j,i}}{r^s_{j,i}} \tag{14}$$

By using this product representation we have introduced an additional degree of freedom for each characteristic function in equation 13. This follows from equation 14 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 $r^s_{j,i}$. For simplicity reasons 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 now 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. This implies that we will not be able to derive quantities such as absolute power of a system or absolute size of a code but only relative ratios between systems and between codes.

Analyzing the full matrix of results $T_{s,c}$ we can now fit values to the individual $t^{s,c}_{j,i}$. 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. The same basic timing model (equation 14) and normalization is used for all system and code parameters which enables fair comparisons between systems and codes.

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. It also implies that fitting experimental data not necessarily lead to an overall model with the same low values of the Sum of Square of the Errors (SSE) as we saw by fitting individual models to the observations for each pair of system and code. We have to expect that the SSE values increase and only by the amount of increase we can judge on the quality of our assumption.

6 Results for the combined model

We now fit all possible timing models of the form of equation 14 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 now compare the results show in table 4 to the results for fitting individual models in table 3.

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 4. The SSE values for the best 5 two function models using the product representation from equation 12.

The values of SSE which characterize the unexplained sum of square are higher for the combined model but this was to expected as we now have only 14 free parameters instead of 32. The absolute increase compared to the Total Sum of Squares (SST) is quite small for each problem size class. This is a first strong confirmation that our factorization assumption from equation 12 works quite well.

We analyze and 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 less and less limited parallel execution.

In table 5 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 11. This means that system parameters are scaled by the peak performance and code parameters by the single processor floating point operation count. The parameters 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 quotients 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 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.

Now we have to check the statistical quality of the obtained fits. The correlation matrix of the parameters for the three models shows 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

Parameter	Class A		
	$u_1 = 1/p$		
r_1^{IBMSP2}	1.	1.	1.
$r_1^{Paragon}$	0.4055 ± 0.0257	0.3704 ± 0.0316	0.3650 ± 0.0406
$r_1^{CrayT3D}$	0.4249 ± 0.0243	0.4103 ± 0.0311	0.4655 ± 0.0467
$r_1^{PCArray}$	0.4822 ± 0.0313	0.4082 ± 0.0372	0.3715 ± 0.0406
w_1^{BT}	4.5723 ± 0.2967	4.0087 ± 0.3646	3.7135 ± 0.4162
w_1^{LU}	4.0230 ± 0.2491	3.5062 ± 0.2973	3.4295 ± 0.3668
w_1^{SP}	5.3556 ± 0.3079	4.1202 ± 0.3373	3.5209 ± 0.3887
w_1^{MG}	4.6719 ± 0.2928	3.9703 ± 0.3467	3.4080 ± 0.3729
	$u_2 = 1$	$u_2 = 1/\sqrt{p}$	$u_2 = log(p)/p$
r_2^{IBMSP2}	1.	1.	1.
$r_2^{Paragon}$	1.5382 ± 0.3595	1.2210 ± 0.2331	1.0122 ± 0.2003
$r_2^{CrayT3D}$	1.9133 ± 0.2657	1.1064 ± 0.1258	0.6986 ± 0.0791
$r_2^{PCArray}$	2.3510 ± 3.6283	3.8782 ± 4.9177	4.0372 ± 3.7810
w_2^{BT}	0.0125 ± 0.0055	0.1716 ± 0.0561	0.4799 ± 0.1143
w_2^{LU}	0.0327 ± 0.0067	0.3194 ± 0.0637	0.5768 ± 0.1257
w_2^{SP}	0.0665 ± 0.0072	0.7525 ± 0.0705	1.5352 ± 0.1540
w_2^{MG}	0.0140 ± 0.0066	0.2350 ± 0.0667	0.7503 ± 0.1323

Parameter	Class B		
	$u_1 = 1/p$		
r_1^{IBMSP2}	1.	1.	1.
$r_1^{Paragon}$	0.4007 ± 0.03295	0.3816 ± 0.0381	0.3810 ± 0.0420
$r_1^{CrayT3D}$	0.4790 ± 0.03536	0.5406 ± 0.0533	0.6696 ± 0.1024
$r_1^{PCArray}$	0.4565 ± 0.03413	0.4521 ± 0.0408	0.4455 ± 0.0416
w_1^{BT}	4.8398 ± 0.38984	4.5633 ± 0.4261	4.3062 ± 0.4130
w_1^{LU}	4.2274 ± 0.31716	4.2681 ± 0.3861	4.3097 ± 0.4034
w_1^{SP}	5.7293 ± 0.40238	5.1657 ± 0.4597	4.9873 ± 0.4747
w_1^{MG}	4.0487 ± 0.32420	3.4547 ± 0.3751	3.1617 ± 0.3677
	$u_2 = 1$	$u_2 = 1/\sqrt{p}$	$u_2 = log(p)/p$
r_2^{IBMSP2}	1.	1.	1.
$r_2^{Paragon}$	6.5208 ± 18.07674	2.9015 ± 3.8635	1.8887 ± 1.8287
$r_2^{CrayT3D}$	1.0531 ± 0.32794	0.5371 ± 0.1433	0.3346 ± 0.0878
$r_2^{PCArray}$	0.1823 ± 0.08624	0.3431 ± 0.1263	0.4218 ± 0.1318
w_2^{BT}	0.0036 ± 0.00496	0.0893 ± 0.0435	0.2713 ± 0.0857
w_2^{LU}	0.0074 ± 0.00557	0.0594 ± 0.0473	0.1576 ± 0.0871
w_2^{SP}	0.0291 ± 0.00845	0.3063 ± 0.0811	0.6151 ± 0.1553
w_2^{MG}	0.0199 ± 0.00713	0.2595 ± 0.0745	0.5476 ± 0.1429

Table 5. 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.

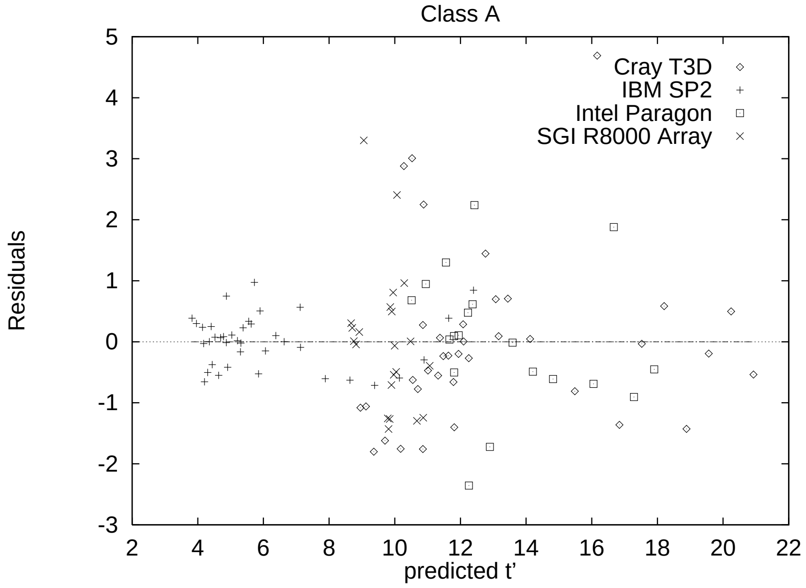

Fig. 1. Distribution of the residuals over the predicted values for the model build with $1/p$ and $1/\sqrt{p}$ for problem size class A.

some outliers at the higher end of the curve. As it is not feasible to visualize the amount of data involved in the whole analysis we present as only figures the values of the residuals versus the predicted values for one of the models and problem size class A in figure 1. 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 physically motivated transformations solves the scale problem of the measured times.
- 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.
- Limitations of the model are strongly varying measurements on the Cray T3D and strong effects of the hierarchical architecture for the SGI R8000 Cluster.

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, Moffett Field, CA 94035, January 1991.
2. D. Bailey, T. Harris, W. Saphir, R. van der Wjingaart, A. Woo and M. Yarrow. The NAS parallel benchmarks 2.0. Technical Report NAS-95-020, NASA Ames Research Center, Moffett Field, CA 94035, December 1995.
3. W. Saphir, A. Woo and M. Yarrow. The NAS parallel benchmarks 2.1 Results. Technical Report NAS-96-01, NASA Ames Research Center, Moffett Field, CA 94035, August 1996.
4. 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.
5. Strohmaier, Erich. Using Computational Similarity to Analyze the Performance Data of the NAS Parallel Benchmarks. Technical Report 44, Rechenzentrum der Universitaet Mannheim, April ,1995,
6. Raj Jain. The Art of Computer Systems Performance Analysis. Wiley, 1991
7. Strohmaier, Erich. Extending the Concept of Computational Similarity for Analyzing Complex Benchmarks. Technical Report 43, Rechenzentrum der Universitaet Mannheim, April ,1995,
8. Vipin Kumar et al.. Introduction to Parallel Computing: Design and analysis of parallel algorithms. Benjamin/Cummings, 1994.
9. R.W. Hockney. The Science of Computer Benchmarking. SIAM, Philadelphia, 1996.
10. 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

This article was processed using the LaTeX macro package with LLNCS style