

Using Kernel Couplings to Predict Parallel Application Performance

Valerie Taylor, Xingfu Wu, Jonathan Geisler

Department of Electrical and Computer Engineering, Northwestern University, Evanston IL 60208

Email: {taylor, wuxf, geisler}@ece.northwestern.edu

Rick Stevens

Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439

Email : stevens@mcs.anl.gov

Abstract

Performance models provide significant insight into the performance relationships between an application and the system used for execution. The major obstacle to developing performance models is the lack of knowledge about the performance relationships between the different functions that compose an application. This paper addresses the issue by using a coupling parameter, which quantifies the interaction between kernels, to develop performance predictions. The results, using three NAS Parallel Application Benchmarks, indicate that the predictions using the coupling parameter were greatly improved over a traditional technique of summing the execution times of the individual kernels in an application. In one case the coupling predictor had less than 1% relative error in contrast the summation methodology that had over 20% relative error. Further, as the problem size and number of processors scale, the coupling values go through a finite number of major value changes that is dependent on the memory subsystem of the processor architecture.

1. INTRODUCTION

Performance models provide significant insight into the performance relationships between an application and the system used for execution. In particular, models can be used to predict the relative performance of different systems used to execute an application or to explore the performance impact of the use of different algorithms to solve a given task. While there are different classes of models, e.g., analytical, simulation-based and statistical, this paper is focused on the class of analytical models because of the ease for which one can use such models to predict performance. The problem with analytical models, however, is with the time required to develop “good” models in the sense of being within say 15% of the actual

execution time. While significant work has been done with developing good analytical models for relatively small kernels (e.g., matrix-matrix multiply, FFT, conjugate gradient solver), very little work has been done in terms of developing a composition algebra that indicates how to combine the kernel models to represent the performance of an application. This paper addresses the issue of a *composition algebra* with the use of the *coupling* parameter, which quantifies the interaction between kernels.

Performance *coupling* refers to the effect that *kernel i* has on *kernel j* in relation to running each kernel in isolation. The two kernels can correspond to adjacent kernels in the control flow of the application or a chain of three or more kernels. In our previous work we used coupling to identify parts of the application that required performance improvement [GE99, GT99]. The coupling value provided insight into where further algorithm and code implementation work was needed to improve performance, in particular the reuse of data between kernels. The previous work was extended in [TG01] to demonstrate how the coupling values of adjacent pairs of kernels were used to develop analytical models for one code, an FFT. This paper, however, explores three different aspects of coupling:

- 1) use of coupling values for chains of three or more kernels to determine how to combine the models of the different kernels,
- 2) how the coupling values change with scaling of the problem size, and
- 3) how the coupling values change with the scaling of the number of processors.

We conducted experiments using the coupling parameter to predict the performance of the NAS Parallel Benchmarks, specifically BT, SP, and LU. The resultant predictions were greatly improved over a traditional technique of summing the execution times of the individual kernels in an application. In one case the coupling predictor had less than 1% relative error in contrast the summation methodology that had over 20%

relative error. In all cases, the coupling predictor was much better than the summation methodology. Further, as the problem size and number of processors scale, the coupling values go through a finite number of major value changes that is dependent on the memory subsystem of the processor architecture.

The remainder of the paper is outlined as follows. Section 2 provides some background on performance coupling followed by a description of how we extend the work in Section 3. The experimental results are given in Section 4. Section 5 discusses related work with the paper summarized in Section 6.

2. BACKGROUND

The coupling parameter, C_{ij} , quantifies the interaction between adjacent kernels in an application. In this work, a kernel is a unit of computation that denotes a logical entity within the larger context of an application. The unit may be a loop, procedure, or file depending on the level of granularity of detail that is desired from the measurements. To compute the parameter C_{ij} , three measurements must be taken:

- P_i is the performance of *kernel i* alone,
- P_j is the performance of *kernel j* alone, and
- P_{ij} is the performance of *kernels i* and *j* (assuming *kernel i* immediately precedes *kernel j*) in the application.

These measurements are done in the sequence determined by the application. In particular, a measurement is obtained by placing a given kernel or pair of kernels into a loop, such that the loop dominates the application execution time. Then the time required for the application, beyond the given kernel or pair of kernels, is subtracted such that the resultant time reflects that of only the given kernel or pair of kernels.

In general, the value C_{ij} is equal to the ratio of the measured performance of the pair of kernels to the expected performance resulting from combining the isolated performance of each kernel. Since C_{ij} is the measurement of interaction between kernels, we compute it as the ratio of the actual performance of the kernels together to that of no interaction, as given below:

$$C_{ij} = \frac{P_{ij}}{P_i + P_j} \quad (1)$$

For the case of a chain of kernels, we define S as the set of kernels to be measured. We measure the performance of the kernels independently (P_k for every *kernel k* in the set S), and the performance of the kernels together (P_S) to compute the coupling parameter C_S . The

equation for the coupling for a chain of kernels is given below:

$$C_S = \frac{P_S}{\sum_{k \in S} P_k} \quad (2)$$

For the case of two kernels, $C_S = C_{ij}$.

The summation of the isolated performance is applicable to performance metrics such as execution time and cache misses. The summation, however, is not applicable to all performance metrics, such as floating point operations per second (flop/s); a weighted average would be used in this case. Different performance metrics may require different mathematical formulations for combining the isolated performance.

It should be noted that interactions between all pairs or chains of kernels are *not* necessary. The value, C_{ij} , represents the direct interaction between two adjacent or chain of kernels in an application (i.e., in the sense of the control flow of the application). For example, for each unique application control path that has N kernels, only $(N-1)$ pair-wise interactions are measured.

We group the parameters into three sets:

- $C_S = 1$ indicates no interaction between the chain of kernels, yielding no change in performance.
- $C_S < 1$ indicates a performance gain, resulting from some resource(s) being shared between the kernels (i.e., constructive coupling).
- $C_S > 1$ indicates a performance loss, resulting from the kernels interfering with each other (i.e., destructive coupling).

3. EXTENDING THE COUPLING CONCEPT

The major obstacle to developing performance models of an application is the lack of knowledge about the performance relationships between the different functions that compose an application. Further, it is necessary to capture this relationship as a coefficient that can be used in an equation. For example, assume we have an application that is composed of *kernelA* followed by *kernelB*, and the application iterates on a loop that contains these two functions until some condition is satisfied. Also assume that we have manually analyzed these two functions such that we have *modelA* and *modelB*, analytical models of the two kernels. A *composition algebra* would provide the coefficients of the relationships between the two kernels, represented below where T is the execution time of the application:

$$T = \mathbf{h} \text{ modelA} + \mathbf{k} \text{ modelB}$$

The coefficients \mathbf{h} and \mathbf{k} represent the performance relationships between the two kernels that identify how they should be combined to reflect the performance of the application. In this section, we describe how we extend our previous work on coupling to quantify the performance relationships between the kernels to represent the performance of an application.

Assume that an application has four kernels (A, B, C, D) that are executed together in one loop. Let E_A , E_B , E_C and E_D represent the analytical models of each of the respective kernels; these models include the number of times the kernel is executed in the application. The equation for the execution time of the application is given as follows:

$$T = \alpha E_A + \beta E_B + \gamma E_C + \delta E_D \quad (3)$$

To determine the values for the coefficients ($\alpha, \beta, \gamma, \delta$), we use a weighted average of the coupling values associated with a given kernel. Recall that the pairwise coupling value indicates the interaction between the pair of kernels. This value is weighted by the fraction of the execution time attributed to the pair of kernels. The weight is needed such that a large coupling value for a pair of kernels that attribute very little to the execution time results in an appropriate valued coefficient.

For the case of the four kernels (A, B, C, D) and using pairwise coupling values, the coefficients have the following values:

$$\alpha = \frac{[(C_{AB} * P_{AB}) + (C_{DA} * P_{DA})]}{(P_{AB} + P_{DA})}$$

$$\beta = \frac{[(C_{AB} * P_{AB}) + (C_{BC} * P_{BC})]}{(P_{AB} + P_{BC})}$$

$$\gamma = \frac{[(C_{BC} * P_{BC}) + (C_{CD} * P_{CD})]}{(P_{BC} + P_{CD})}$$

$$\delta = \frac{[(C_{CD} * P_{CD}) + (C_{DA} * P_{DA})]}{(P_{CD} + P_{DA})}$$

For example, the coefficient, α , for the kernel A, includes the coupling values that include kernel A, C_{AB} and C_{DA} . These values are weighted by the ratio of the execution time attributed to each pair to the summation of the execution times of the two pairs. Similar equations result for the other coefficients.

The above example generates the coefficients using the pairwise coupling values. The coefficients can also be generated using the coupling values for a chain of kernels. For example, given an application with the flow control of $A \rightarrow B \rightarrow C \rightarrow D$, for a chain of length three, the kernel A

is included in the coupling values -- $C_{ABC}, C_{CDA}, C_{DAB}$. To generate the coefficient for kernel A, α , we use a weighted average similar to what was used for the pairwise coupling values. The equations for the coefficients using a kernel chain of length three are given below:

$$\alpha = \frac{[(C_{ABC} * P_{ABC}) + (C_{CDA} * P_{CDA}) + (C_{DAB} * P_{DAB})]}{(P_{ABC} + P_{CDA} + P_{DAB})}$$

$$\beta = \frac{[(C_{ABC} * P_{ABC}) + (C_{BCD} * P_{BCD}) + (C_{DAB} * P_{DAB})]}{(P_{ABC} + P_{BCD} + P_{DAB})}$$

$$\gamma = \frac{[(C_{ABC} * P_{ABC}) + (C_{BCD} * P_{BCD}) + (C_{CDA} * P_{CDA})]}{(P_{ABC} + P_{BCD} + P_{CDA})}$$

$$\delta = \frac{[(C_{BCD} * P_{BCD}) + (C_{CDA} * P_{CDA}) + (C_{DAB} * P_{DAB})]}{(P_{BCD} + P_{CDA} + P_{DAB})}$$

The use of the weighted average assumes that when the coupling values were generated, the number of times a kernel is executed is fixed. Further, it is assumed that the different runs needed to generate all of the coupling values have the same input. As to which group of equations will lead to the best prediction, is an area of future work.

4. CASE STUDIES: Three NAS Parallel Application Benchmarks

In this section, we present experimental results focused on the three different aspects of coupling: (1) use of coupling values for chains of two or more kernels to determine how to combine the models of the different kernels, (2) how the coupling values changes with scaling of the problem size, and (3) how the coupling values changes with the scaling of the number of processors. The experiments use three application benchmarks of the NAS Parallel Benchmarks (NPB) suite, BT, SP and LU [BH95], executed on the IBM SP at Argonne National Laboratory. This machine consists of 80 processors, each with 120 MHz P2SC CPUs; the system includes approximately 40 GB of memory, 1TB of on-line storage and 60TB off-line storage.

4.1 NPB: BT

BT (Block Tridiagonal) is an application benchmark, which solves three sets of uncoupled systems of equations, first in the x dimension, then in the y dimension, and finally in the z dimension. These systems are block tri-diagonal with 5x5 blocks. We divided the application benchmark into seven kernels:

- 1) **INITIALIZATION**: Initializes values.
- 2) **COPY FACES**: Phase one computation of the right hand side.
- 3) **X SOLVE**: Solves the problem in the x dimension.
- 4) **Y SOLVE**: Solves the problem in the y dimension.
- 5) **Z SOLVE**: Solves the problem in the z dimension.
- 6) **ADD**: Updates values.
- 7) **FINAL**: Verifies the solution integrity.

The BT control flow entails executing steps 2 through 6 in a loop. Hence, we focus on obtaining coupling values for the five main kernels. The parallel BT benchmark, which uses MPI [GL94], requires that the number of processors is a square. The data sets for BT benchmark are showed in Table 1. We present the results organized by data set size. Further, for the sake of brevity, we present only the coupling values corresponding to the length of the chain of kernels that produced best predictions. In all cases, with the exception of the few cases discussed in the paper, the coupling predictors performed better than the summation predictor.

| BT Data Set | Size |
|-------------|--------------|
| S | 12 x 12 x 12 |
| W | 32 x 32 x 32 |
| A | 64 x 64 x 64 |

Table 1: Data sets used with the NPB BT.

The kernels **INITIALIZATION** and **FINAL** are called once for Class S, W and A. The kernels **COPY FACES**, **X SOLVE**, **Y SOLVE**, **Z SOLVE**, and **ADD** are called 60 times for Class S, and 200 times for Class W and A.

Summation is the methodology of accumulating the total time of each individual kernel to approximate the total execution time. Let T_{init} , T_{c-f} , T_{x-s} , T_{y-s} , T_{z-s} , T_{add} and T_{final} be the average execution time of the kernel **INITIALIZATION**, **COPY FACES**, **X SOLVE**, **Y SOLVE**, **Z SOLVE**, **ADD** and **FINAL**, respectively. The average execution time for each kernel is obtained by running the kernel 50 times. Then the following equation predicts the execution time of BT for Class S using the summation methods:

$$\text{Summation} = T_{init} + 60 * (T_{c-f} + T_{x-s} + T_{y-s} + T_{z-s} + T_{add}) + T_{final}$$

For the case of datasets W and A, a similar equation is used with the replacement of 60 by 200 corresponding to the number of times the loop is executed for the given dataset sizes.

4.1.1 Small Dataset (S)

Table 2a provides the coupling values obtained for the pairs of kernels within BT. The general trend is that the coupling values get larger as the number of processors increase (the exception is the {Add, Copy Faces} pair using 9 processors). We believe the reason for this trend is threefold:

- 1) As the number of processors increases, the amount of data computed per processor decreases. This allows more of the data to reside in the caches allowing kernels to be affected by the sharing of data through the cache. The sharing can be either destructive or constructive.
- 2) As the number of processors increase, the number of messages in the communication system at any time increases (higher coupling values) while the size of each message decreases (lower coupling values).
- 3) As the number of processors increase, the load imbalance can increase. Some processors may have to wait for relatively long periods of time while others finish their computations. This will result in higher coupling values.

The data indicates that the number of messages and load balancing issues are affecting the coupling more than the message sizes and cache effects. The anomalies occur when the caching or message lengths are significant. This is true for the copy faces kernel because this kernel has a lot of data movement, so the cache misses and message lengths have a greater effect as compared to other kernels that compute data values in-place.

Table 2a: Coupling values for BT two kernels with Class S

| Kernel Pair | Coupling Value | | |
|---------------------|----------------|--------------|---------------|
| | 4 processors | 9 processors | 16 processors |
| Copy_Faces, X_Solve | 0.9522 | 1.1639 | 1.0121 |
| X_Solve, Y_Solve | 1.0189 | 1.0559 | 1.2828 |
| Y_Solve, Z_Solve | 0.9705 | 0.8480 | 1.0781 |
| Z_Solve, Add | 1.0082 | 0.9015 | 1.1528 |
| Add, Copy_Faces | 0.8259 | 2.5180 | 1.0314 |

Table 2b: Comparison of execution times for BT with Class S

| Prediction Type | Execution Time in seconds (% relative error) | | |
|---------------------|--|-------------------------|-------------------------|
| | 4 processors | 9 processors | 16 processors |
| Actual | 1.4576 | 1.1817 | 0.8161 |
| Summation | 1.2032 (17.45 %) | 0.7332 (37.95 %) | 0.5161 (36.76 %) |
| Coupling: 2 kernels | 1.1791 (19.11 %) | 0.7507 (36.47 %) | 0.5746 (29.58 %) |

Table 2b gives the comparison of execution for the different prediction methods. The first column lists the prediction type where *Actual* is the measured execution time, *Summation* is the methodology of accumulating the total time of each individual kernel, and the last row uses the coupling parameter with given number of kernels in the chain. The remaining three columns list the execution time in seconds and, for the case of the predictions, the relative error from the measured execution time.

For the small dataset size, we see that the prediction values in Table 2b are not very close to the actual execution times. We believe that this occurs because the predicted execution time is so small, that measuring errors get magnified quickly. The best predictor is boldfaced in Table 2b, and we can see that the prediction using two kernels gives better results for larger numbers of processors, while the summation does best for 4 processors. The average relative error for the summation method is 30.72%, while the average relative error for the two kernel predictor is 28.39%.

4.1.2 Workstation Dataset (W)

Table 3a demonstrates that there is a large amount of constructive coupling that takes place when considering three kernels. All coupling values are below 0.85. This should result in execution times less than the summation of the individual kernels as indicated below in Table 3b. Further, as the number of processors increases, the coupling value changes very little. This occurs because the problem size per process is such that the data is too large for the first-level cache, but fits within the second-level cache. Hence, we do not see the effects as given above for the smaller data set size.

Table 3b shows that predictors that take coupling into account give more accurate results than summing the individual kernels' execution times in all cases. The three kernel predictors have an average relative error of 1.42%, while the more traditional summation methodology has an average relative error of 22.42%. This is a significant improvement.

Table 3a: Coupling values for BT three kernels with Class W

| Three Kernels | Coupling Value | | | |
|------------------------------|----------------|--------------|---------------|---------------|
| | 4 processors | 9 processors | 16 processors | 25 processors |
| Copy_Faces, X_Solve, Y Solve | 0.7663 | 0.7427 | 0.7665 | 0.7941 |
| X_Solve, Y_Solve, Z Solve | 0.8058 | 0.8013 | 0.7971 | 0.8278 |
| Y_Solve, Z_Solve, Add | 0.8271 | 0.8010 | 0.8475 | 0.8423 |
| Z_Solve, Add, Copy Faces | 0.7338 | 0.7986 | 0.7621 | 0.8165 |
| Add, Copy_Faces, X Solve | 0.8435 | 0.7631 | 0.7959 | 0.8250 |

Table 3b: Comparison of execution times for BT with Class W using three kernels

| Prediction Type | Execution Time in Seconds (% Relative Error) | | | |
|----------------------------|--|-------------------------|------------------------|------------------------|
| | 4 processors | 9 processors | 16 processors | 25 processors |
| Actual | 28.1184 | 14.6031 | 9.8032 | 7.5263 |
| Summation | 34.8469 (23.93 %) | 18.1717 (24.44 %) | 12.0792 (23.22 %) | 8.8887 (18.10 %) |
| Prediction using 3 kernels | 27.7961 (1.15 %) | 14.2328 (2.54 %) | 9.6097 (1.97 %) | 7.3004 (3.00 %) |

Table 4a: Coupling values for BT four kernels with Class A

| Four Kernels | Coupling Value | | | |
|---------------------------------------|----------------|--------------|---------------|---------------|
| | 4 processors | 9 processors | 16 processors | 25 processors |
| Copy Faces, X Solve, Y Solve, Z Solve | 0.8951 | 0.7623 | 0.7899 | 0.7991 |
| X Solve, Y Solve, Z Solve, Add | 0.7957 | 0.7950 | 0.8137 | 0.8316 |
| Y Solve, Z Solve, Add, Copy Faces | 0.9282 | 0.7860 | 0.7917 | 0.8037 |
| Z Solve, Add, Copy Faces, X Solve | 0.9732 | 0.7852 | 0.8216 | 0.8546 |
| Add, Copy Faces, X Solve, Y Solve | 0.8948 | 0.7550 | 0.7628 | 0.7498 |

Table 4b: Comparison of execution times for BT with Class A

| Prediction Type | Execution Time in Seconds (% Relative Error) | | | |
|---------------------|--|------------------------------------|------------------------------------|------------------------------------|
| | 4 processors | 9 processors | 16 processors | 25 processors |
| Actual | 572.8050 | 263.6280 | 156.3300 | 107.7080 |
| Summation | 633.7375 (10.64 %) | 335.5829 (27.29 %) | 196.6579 (25.80 %) | 132.9664 (23.45 %) |
| Coupling: 4 kernels | 562.9191 (1.73 %) | 260.8929 (1.04 %) | 156.8362 (0.32 %) | 107.6433 (0.06 %) |

4.1.3 Large Dataset A

Table 4a provides the coupling values for four kernel chains as this was the coupling case that provided the best predictions. The data indicates that for most four kernel pairs, the coupling value decreases when going from 4 to 9 processors and then has very little change beyond 9 processors. For these cases where some change in coupling value occurs, the coupling values are in the range of 0.9 indicating very little constructive coupling. This occurs because for the case of 4 processors, the problem size per processor is such that the data for the four kernel case is definitely beyond the caches. Hence, the coupling between the four kernels is such that there is little constructive coupling. However, as the problem size decreases per processor, we see some constructive coupling with values in the range of 0.8.

Table 4b indicates that the four kernel predictor has the best accuracy overall with an average relative error of 0.79%. We are starting to see that as the dataset increases we need to consider more kernels when computing coupling. This is specific to the BT application and it is an open question whether this holds for all applications.

4.1.4 Summary

The BT application was modeled very well when we considered three and four kernels coupling values. As the dataset size increased, we saw the accuracy of these two

predictors increased. Also, as the number of processors increased, we saw the accuracy of these two coupling predictors increased. The summation methodology and the two kernel coupling predictor did not seem to change very much with regards to these two factors. These results demonstrate that using coupling parameter can greatly increase the accuracy of execution time predictions. Further, as the number of processors increases, we see that the coupling value changes with respect to the effects of the cache. We find that the coupling values go through a finite number of transitions as the problem size scales.

4.2 NPB: SP

SP (Scalar Pentadiagonal) is an application benchmark, which solves three sets of uncoupled systems of equations, first in the x, then in the y, and finally in the z dimension. These systems are scalar pentadiagonal. We divided the application benchmark into eight kernels:

- 1) INITIALIZATION: Initializes values.
- 2) COPY FACES: Phase one computation of the right hand side.
- 3) TXINVR: Phase two computation of the right hand side.
- 4) X SOLVE: Solves the problem in the x dimension.
- 5) Y SOLVE: Solves the problem in the y dimension.
- 6) Z SOLVE: Solves the problem in the z dimension.
- 7) ADD: Updates values.
- 8) FINAL: Verifies the solution integrity.

The SP control flow entails executing steps 2 through 7 in a loop. Hence, we focus on obtaining coupling values for the six main kernels. The parallel SP benchmark requires that the number of processors is a square. The data sets for SP benchmark are given in Table 5. For brevity, we only include the prediction tables for the SP application. We used the same methods to compute the predictions and saw similar trends in the coupling values in SP as with BT.

| SP Data Set | Size |
|-------------|-----------------|
| W | 36 x 36 x 36 |
| A | 64 x 64 x 64 |
| B | 102 x 102 x 102 |

Table 5: Data sets used with the NPBS

Table 6a: Comparison of execution times for SP with Class W

| Prediction Type | Execution Time (seconds) (% Relative Error) | | | |
|-----------------------|---|-----------------------------------|-----------------------------------|-----------------------------------|
| | 4 processors | 9 processors | 16 processors | 25 processors |
| Actual Execution Time | 81.7139 | 40.4299 | 26.8212 | 19.5731 |
| Summation | 104.2775 (27.61%) | 46.82385 (15.81%) | 30.23738 (12.74%) | 21.06709 (7.63%) |
| Coupling: 4 kernels | 82.93803 (1.50%) | 40.33829 (0.23%) | 26.25598 (2.11%) | 19.05079 (2.67%) |
| Coupling: 5 kernels | 81.86098 (0.18%) | 40.05919 (0.92%) | 26.67478 (0.55%) | 19.79375 (1.13%) |

Table 6b: Comparison of execution times for SP with Class A

| Prediction Type | Execution Time (seconds) (% Relative Error) | | | |
|-----------------------|---|-----------------------------------|-----------------------------------|-----------------------------------|
| | 4 processors | 9 processors | 16 processors | 25 processors |
| Actual Execution Time | 478.828 | 205.970 | 129.546 | 84.9322 |
| Summation | 618.0988 (29.09%) | 247.3636 (20.10%) | 152.9177 (18.04%) | 97.60911 (14.93%) |
| Coupling: 4 kernels | 457.1872 (4.52%) | 211.0591 (2.47%) | 129.5085 (0.02%) | 84.20481 (0.86%) |
| Coupling: 5 kernels | 470.0681 (1.83%) | 208.1988 (1.08%) | 131.2545 (1.32%) | 85.33942 (0.48%) |

Table 6c: Comparison of execution times for SP with Class B

| Prediction Type | Execution Time (seconds) (% Relative Error) | | | |
|-----------------------|---|-----------------------------------|-----------------------------------|-----------------------------------|
| | 4 processors | 9 processors | 16 processors | 25 processors |
| Actual Execution Time | 2062.01 | 770.424 | 471.2020 | 304.9790 |
| Summation | 2538.033 (23.09%) | 928.3681 (20.50%) | 562.0345 (19.34%) | 361.7253 (18.61%) |
| Coupling: 4 kernels | 2075.067 (0.63%) | 778.0906 (1.00%) | 478.4723 (1.54%) | 310.6063 (1.85%) |
| Coupling: 5 kernels | 2099.987 (1.84%) | 781.0699 (1.38%) | 475.9352 (1.00%) | 310.3162 (1.75%) |

Table 6a shows that using coupling performed better than the summation method, which had an average relative error of 15.95%, while the coupling predictor had an average relative error of 1.63% for 4 kernels coupling, and 0.70% for 5 kernels coupling.

We see similar trends in Tables 6b and 6c as we saw in Table 6a with the coupling predictors noticeably better than the summation methodology. Table 6b shows that using coupling performed better than the summation method, which had an average relative error of 20.54%, while the coupling predictor had an average relative error of 1.97% for 4 kernels coupling, and 1.18% for 5 kernels coupling. In Table 6c, the worst relative error for coupling is 1.85%, while the best relative error for the summation methodology is 18.61%. Again, finite transitions occurred with the coupling values as the number of processors and problem size scale.

4.3 NPB: LU

The LU benchmark requires a power-of-two number of processors. A 2-D partitioning of the grid onto processors occurs by halving the grid repeatedly in the first two dimensions, alternately x and then y, until all power-of-two processors are assigned, resulting in vertical pencil-like grid partitions on the individual processors. The ordering of point based operations constituting the SSOR procedure proceeds on diagonals which progressively sweep from one corner on a given z plane to the opposite corner of the same z plane, thereupon proceeding to the next z plane. Communication of partition boundary data occurs after completion of computation on all diagonals that contact an adjacent partition. This constitutes a diagonal pipelining method. It results in a relatively large

number of small communications of five words each. It is very sensitive to the small-message communication performance.

We have divided the benchmark into ten kernels:

- 1) **INITIALIZATION**: sets all the initial values for various matrices, vectors, and scalars.
- 2) **ERHS**: computes the forcing matrix.
- 3) **SSOR_INIT**: initializes various values for SSOR
- 4) **SSOR_ITER**: performs SSOR iteration
- 5) **SSOR_LT**: performs the lower triangular solution
- 6) **SSOR_UT**: performs the upper triangular solution
- 7) **SSOR_RS**: updates the variables, and computes Newton iteration residuals
- 8) **ERROR**: iterates through the ce matrix
- 9) **PINTGR**: computes the surface integral
- 10) **FINAL**: verifies the solution integrity and cleans up any data structure along with printing out the results of the computation.

The LU control flow entails executing steps 4 through 7 in a loop. Hence, we focus on obtaining coupling values for the four main kernels. The parallel LU benchmark requires that the number of processors is a power of 2. The data sets for LU benchmark are showed in Table 7.

Table 7 Data sets used with the NPB LU

| LU Data Set | Size |
|-------------|-----------------|
| W | 33 x 33 x 33 |
| A | 64 x 64 x 64 |
| B | 102 x 102 x 102 |

Table 8a Comparison of execution times for LU with Class W

| Prediction Type | Execution Time (seconds) (% Relative Error) | | | |
|-----------------------|---|-----------------------------------|-----------------------------------|-----------------------------------|
| | 4 processors | 8 processors | 16 processors | 32 processors |
| Actual Execution Time | 65.0657 | 40.4449 | 20.8262 | 13.2299 |
| Summation | 71.07346 (9.23%) | 40.52920 (0.21%) | 21.74169 (4.40%) | 18.21382 (37.67%) |
| Coupling: 3 kernels | 66.15179 (1.67%) | 40.52396 (0.19%) | 21.35540 (2.54%) | 14.45664 (9.27%) |

Table 8b Comparison of execution times for LU with Class A

| Prediction Type | Execution Time (seconds) (% Relative Error) | | | |
|-----------------------|---|-----------------------------------|-----------------------------------|-----------------------------------|
| | 4 processors | 8 processors | 16 processors | 32 processors |
| Actual Execution Time | 385.424 | 210.414 | 115.930 | 65.97659 |
| Summation | 417.0396 (8.20%) | 218.2716 (3.73%) | 118.4446 (2.17%) | 68.70924 (4.14%) |
| Coupling: 3 kernels | 388.9882 (0.92%) | 208.6102 (0.86%) | 114.7253 (1.04%) | 63.95012 (3.07%) |

Table 8c Comparison of execution times for LU with Class B

| Prediction Type | Execution Time (seconds) (% Relative Error) | | | |
|-----------------------|---|-----------------------------------|-----------------------------------|-----------------------------------|
| | 4 processors | 8 processors | 16 processors | 32 processors |
| Actual Execution Time | 1813.38 | 910.352 | 469.301 | 251.752 |
| Summation | 1873.875 (3.34%) | 933.8349 (2.58%) | 487.1409 (3.80%) | 257.4962 (2.28%) |
| Coupling: 3 kernels | 1818.685 (0.29%) | 906.4860 (0.42%) | 462.5654 (1.44%) | 255.0427 (1.31%) |

Table 8a shows that using coupling performed better than the summation method, which had an average relative error of 12.88%, while the coupling predictor had an average relative error of 3.60%.

We see similar trends in Tables 8b and 8c as we saw in Table 6a with the coupling predictors noticeably better than the summation methodology. Table 8b shows that using coupling performed better than the summation method, which had an average relative error of 4.56%, while the coupling predictor had an average relative error of 1.47%. In Table 8c, the worst relative error for coupling is 1.44%, while the best relative error for the summation methodology is 2.28%.

5. RELATED WORK

Allan Snavely defined a similar concept to coupling as symbiosis [SC99]. His work studied the interaction of two separate programs on a multithreaded machine; in contrast our work focuses on the interactions within an application. The equations used to quantify interaction uses a similar ratio of isolated and simultaneous execution times. His work in [SC00] indicated that dynamic scheduling of multiple highly tuned applications still produces better efficiency on the Tera MTA. We believe our work compliments this work nicely in that we can provide the highly tuned applications to their dynamic scheduler.

Rafael Saavedra did much work characterizing various benchmarks [SS92] by decomposing them into high-level Fortran statements. Saavedra's work demonstrated that measurements of high level constructs can result in

accurate predictions (over 50% of the predictions had less than 10% error) if cache or compiler optimizations are not used. In [SS93], he added terms in his model to account for cache effects. This improved the accuracy of his model to more than 80% of the predictions with less than 10% error. Our work compliments Saavedra's work in quantifying and understanding the interaction between kernels.

6. CONCLUSIONS

We have seen that using performance coupling values can greatly improve the prediction of execution times over a summation methodology. For BT dataset A, the four kernel predictor had an average relative error of 0.79%, while the summation methodology had an average relative error of 21.80%. For the SP dataset A, the four kernel predictor had an average relative error of 1.97%, while the summation methodology had an average relative error of 20.54%. For the LU dataset A, the three kernel predictor had an average relative error of 1.47%, while the summation methodology had an average relative error of 4.56%. Additionally, we often saw the accuracy of all models improve with an increase in number of processors or dataset size. Further, for the BT benchmark, the data indicated that the coupling values go through a finite number of transitions as the problem size and number of processors scale. These results demonstrate that with a few more measurements to obtain coupling parameters, one can create more accurate models of the applications. Future work is focused on determining which coupling

values must be obtained and which values can be reused, thereby reducing the number of needed experiments. In addition future work is focused distributed applications.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the MCS Division of Argonne National Laboratory for the use of the IBM SP for the case studies. This work is supported in part by NASA Ames, NSF NGS grant EIA-9974960, and NSF ITR grants (ITR-0086044 and ITR-0085952).

REFERENCES

- [BH95] D. Bailey, T. Harris, et al., *The NAS Parallel Benchmarks*, Tech. Report NAS-95-020, Dec. 1995. See also <http://science.nas.nasa.gov/Software/NPB/>.
- [GE99] J. Geisler, *Performance Coupling: A Methodology for Analyzing Application Performance using Kernel Performance*, MS Thesis, Northwestern University, March 1999.
- [GL94] W. Gropp, E. Lusk and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MIT Press, Cambridge, MA, 1994.
- [GT99] J. Geisler and V. Taylor, Performance coupling: A methodology for predicting application performance using kernel performance, In *Proc. of the 9th SIAM Conference on Parallel Processing for Scientific Computing*, March 1999.
- [SC99] A. Snively, N. Mitchell, L. Carter, J. Ferrante, and D. Tullsen. Explorations in symbiosis on two multithreaded architectures, In *Proc. Of the 2nd Workshop on Multi-Threaded Execution Architecture and Compilers*, January 1999.
- [SC00] A. Snively and L. Carter. Symbiotic job scheduling on the Tera MTA, In *Proc. Of 3rd Workshop on Multi-Threaded Execution Architecture and Compilers*, January 2000.
- [SS92] R. Saavedra and A. J. Smith. *Analysis of benchmark characteristics and benchmark performance prediction*, Technical Report CSD-92-715, University of California, Berkeley, 1992.
- [SS93] R. Saavedra and A. J. Smith. *Measuring cache and TLB performance and their effect on benchmark run times*, Technical Report CSD-93-767, University of California, Berkeley, 1993.
- [TG01] V. Taylor, X. Wu, J. Geisler, X. Li, Z. Lan, M. Hereld, I. Judson, R. Stevens, "Prophesy: Automating the Modeling Process", *Invited Paper, Third International Workshop on Active Middleware Services 2001*, in conjunction with HPDC-10, August 2001.