PERFORMANCE ANALYSIS OF THE PARALLEL COMMUNITY ATMOSPHERE MODEL (CAM) APPLICATION

A Thesis

by

SAMEH SHERIF SHAWKY SHARKAWI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2006

Major Subject: Computer Science

PERFORMANCE ANALYSIS OF THE PARALLEL COMMUNITY ATMOSPHERE MODEL (CAM) APPLICATION

A Thesis

by

SAMEH SHERIF SHAWKY SHARKAWI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Valerie Elaine Taylor
Committee Members,	Nancy Amato
	Ping Chang
Head of Department,	Valerie Elaine Taylor

August 2006

Major Subject: Computer Science

ABSTRACT

Performance Analysis of the Parallel
Community Atmosphere Model (CAM) Application. (August 2006)
Sameh Sherif Shawky Sharkawi, B.S., The American University in Cairo
Chair of Advisory Committee: Dr. Valerie Elaine Taylor

Efficient execution of parallel applications requires insight into how the parallel system features impact the performance of the application. Significant experimental analysis and the development of performance models enhance the understanding of such an impact. Deep understanding of an application's major kernels and their design leads to a better understanding of the application's performance, and hence, leads to development of better performance models. The Community Atmosphere Model (CAM) is the latest in a series of global atmospheric models developed at the National Center for Atmospheric Research (NCAR) as a community tool for NCAR and the university research community.

This work focuses on analyzing CAM and understanding the impact of different architectures on this application. In the analysis of CAM, kernel coupling, which quantifies the interaction between adjacent and chains of kernels in an application, is used. All experiments are conducted on four parallel platforms: NERSC (National Energy Research Scientific Computing Center) Seaborg, SDSC (San Diego Supercomputer Center) DataStar P655, DataStar P690 and PSC (Pittsburgh Supercomputing Center) Lemieux. Experimental results indicate that kernel coupling gave an insight into many of the application characteristics. One important characteristic of CAM is that its performance is heavily dependent on a parallel platform memory hierarchy; different cache sizes and different cache policies had the major effect on CAM's performance. Also, coupling values showed that although CAM's kernels share many data structures, most of the coupling values are still destructive (i.e., interfering with each other so as to adversely affect performance). The kernel coupling results helps developers in pointing out the bottlenecks in memory usage in CAM. The results obtained from processor partitioning are significant in helping CAM users in choosing the right platform to run CAM.

DEDICATION

To my mother who gave me all the support in the world

ACKNOWLEDGMENTS

I would like to express my gratitude to the people who have supported me in completing this thesis. I am especially thankful to my advisor, Dr. Valerie E. Taylor, who has provided encouragement, insight, and guidance over the course of the work presented in this thesis. I would also like to thank Dr. Xingfu Wu for always providing help, support, and immediate feedback. Also, I would like to give thanks to my colleagues in the research group for helping me whenever I needed them and giving positive feedback on my work. Finally, I would like to thank Dr. Patrick Worley for providing the application and answering questions about it, the PERC group and Dr. David Bailey for providing us with accounts on NERSC Seaborg, San Diego Supercomputing Center for providing us with accounts on DataStar P655 and P690 and for their technical support when needed, and finally Pittsburgh Supercomputing Center for providing us with accounts on Lemieux.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Computational models enable us to continually refine our understanding of earth systems and predict weather and climate. During the past couple of years, severe weather conditions and their disastrous consequences show the extreme need for such models in predicting atmospheric and climate conditions. The Community Atmosphere Model (CAM) is the latest in a series of global atmosphere models developed at the National Center for Atmospheric Research (NCAR) [1]. It was originally developed to simulate general circulation of the atmosphere, and was later modified to work with other components of the climate system model to simulate climate. It was developed in Fortran 90 and has support for shared memory (via OpenMP) and message passing (via MPI). CAM is composed of several hundred files encompassing physics, dynamics and ocean sciences. When running CAM, researchers usually simulate 30 to 40 simulation years on average. For the smallest resolution dataset, to simulate one day requires approximately 42 seconds on 32 Power3 processors; to simulate 30 years requires 120 CPU hours per processor for the smallest dataset. For such large-scale applications, how to understand their performance and point out bottlenecks becomes a major challenge because of the variation of schemes used in communication and computation. Also, the variations of operating systems, machine architectures, compilers and runtime libraries complicate the understanding of such applications' behavior.

This thesis focuses on analyzing the Community Atmosphere Model (CAM) and understanding the performance impact on different architectures. This work investigates the following four aspects of CAM's performance:

- application input: this work uses three different datasets and investigates the effect of dataset (grid) size on the application performance.
- system configuration: this work examines how the number of processors per node impacts the application performance.
- scalability: this work examines the application performance for a fixed problem size with processor scaling.

This thesis follows the style of *ACM Sigmetrics.*

- kernel coupling: this work uses the kernel coupling metric to examine the impact of system parameters on application performance

CAM runs on large scale supercomputers; thus, this analysis will give the researchers a guide on the best configuration for CAM on such systems and paves the way for a yet better design. CAM was executed on four supercomputers: SDSC DataStar P655, SDSC DataStar P690, NERSC Seaborg and PSC Lemieux; a detailed description and comparison among these machines are provided below. Each of these supercomputers has different number of processors per node, different memory hierarchies and different network interconnections; thus, executing CAM with different processor partitioning shows how CAM's behavior is affected with such partitioning and how the difference in memory hierarchy and network interconnections impacts the trend of execution.

Kernel coupling quantifies the interaction between adjacent and chains of kernels in an application [2]. There are four major kernels in CAM: (1) PHYS_PKG that approximates subgrid phenomena such as precipitation processes, clouds, long and short wave radiation, and turbulent mixing, (2) DYN_PKG that advances the evolution equations for the atmospheric flow, (3) P_D_COUPLING which is responsible for converting physics data to dynamics data, and (4) D_P_COUPLING which is responsible for converting data from dynamics to physics. Among these four kernels two dominate the execution, PHYS_PKG and DYN_PKG. In this work, coupling between the four kernels is closely examined and analyzed. The runtimes of each kernel when executed in isolation and runtimes when executed in pairs and chains of three kernels were examined. In addition, the scalability of the overall application in comparison to the scalability of each kernel in isolation is studied. These tests where conducted on the four aforementioned supercomputers, NERSC Seaborg, SDSC DataStar P655, P690 and PSC Lemieux.

CAM supports three dynamical cores: Spectral Eulerian Dynamics, Semi-Lagrangian Dynamics, and Finite Volume Dynamics. This work focuses on the Eulerian Dynamical core. The Spectral Eulerian Dynamics is the default dynamical core for CAM and has been used the longest among other dynamical cores by the scientific research community. There are three standard configurations that are available on CAM website for the Eulerian Dynamical core with spectral resolutions T31, T42 and T85.

1. *T31:* 96 Longitude x 48 Latitude. This takes 48 timesteps to simulate 1 day.
2. *T42:* 128 Longitude x 64 Latitude. This takes 72 timesteps to simulate 1 day.
3. *T85:* 256 Longitude x 128 Latitude. This takes 145 timesteps to simulate 1 day.

Although the three datasets have a third dimension which is the level, but this can be chosen to be either 26 or 30 during the configuration. Through out all the tests concerned in this work, the level was set to be 26. In this way, it is guaranteed in each run that the PHYS_PKG will be called.

Kernel coupling gave us an insight about many of the application characteristics. One important characteristic of CAM is that its performance is heavily dependent on parallel platform memory hierarchy. For example, all the coupling values show that Lemieux, with a slower processor than DataStar but with a larger L1 and L2 cache sizes, experience better coupling than DataStar. Also, coupling values showed that although CAM's kernels share many data structures, most of the coupling values are still destructive (i.e., interfering with each other so as to adversely affect performance). This is due to the large sizes of these data structures and the way CAM loops are designed.

In processor partitioning, although the intranode bandwidth is much higher than the internode bandwidth, CAM's runtime was better when less than half of the maximum number of processors per node are used. This, also, emphasizes CAM's heavy reliance on memory and that the intensive use of memory in computation reduces the intranode (i.e., within a node) bandwidth significantly.

2. BACKGROUND

Performance analysis and prediction provide significant insight into the performance relationships between an application and the system used for execution. The major obstacle to correctly understand an application behavior and performance is the lack of knowledge about the performance relationships between the different functions that compose an application [6]. Understanding such relationships assists in deriving performance models that help in predicting and understanding an application behavior.

Kernel coupling refers to the effect that *kernel i* has on *kernel j* in relation to running each kernel in isolation. The two kernels can be adjacent kernels in the control flow of the application or a chain of three or more kernels [5]. In this work, kernel coupling will be used to identify four major points:

- how the coupling values change with scaling of the problem size;
- how the coupling values change with the scaling of the number of processors;
- how coupling values change with the system architecture; and
- how coupling values change with the application runtime.

2.1 Testbeds

In this work, all CAM runs and tests were performed on four supercomputers: SDSC DataStar P655 and P699, NERSC Seaborg and PSC Lemieux. Table 1 shows a detailed comparison for these machines. As it is clear in the table, P655, P690 and Seaborg have the same Operating System (AIX), but they have different processor speeds and different memory hierarchy. For that reason, we chose the PSC Lemieux machine as the 4th testbed in order to have a different operating system and different runtime libraries to compare with. The four machines had support for message passing (MPI) and shared memory (OpenMP). In addition, the four machines had the libraries needed to run CAM installed and specifically the Network Common Data Form (NetCDF). NetCDF is an interface for array-oriented data access and a library that provides an implementation of the interface. The netCDF library also defines a machine-independent format for representing scientific data. Together, the interface, library, and format support the creation, access, and sharing of scientific data [4]. For the three IBM machines, IBM XL Fortran compiler was used. Although the machines have different compilers installed on the machines, the choice of the XL FORTRAN was due to its capability of optimizing the code to the best extent on the IBM machines. On the other hand, f90 was used on Lemieux. For each of these machines, there were certain batch scripts that were written to configure the runtime environment for CAM. The most important parameter that was needed by CAM on all platforms was setting the Stack Size to maximum value in order for CAM to run without crashing. This is due to CAM's intensive memory requirements which will be discussed in details in CAM Description and CAM Analysis sections.

Table 1: Testbeds Comparison

Configurations	**SDSC DataStar P655**	**SDSC DataStar P690**	**NERSC Seaborg**	**PSC Lemieux**
Number of Nodes	176	7	416	750
CPUs per Node	8	32	16	4
CPU type	1.5 GHz PPC4	1.7 GHz PPC4	375 MHz PPC3	1 GHz Alpha
CPU Peak Speed	6.0 GFlops	6.8 GFlops	1.5 GFlops	600 MFlops
Memory per Node	16GB	128GB	16-64GB	4 GB
L1 Cache	64/32 KB 2-way/ Direct Mapped	64/32 KB 2-way/ Direct Mapped	64/32 KB 128-way set associative	64/64 KB 2-way set associative
L2 Cache	1.5MB On Chip shared between 2 cores	1.5MB On Chip shared between 2 cores	8MB Off Chip per one core	8MB On Chip
L3 Cache	128MB	128MB	N/A	N/A
Network	Federation	Federation	Colony	Quadrics
OS	AIX	AIX	AIX	Tru64 Unix

2.2 CAM

This section describes CAM and has a brief scientific explanation for the physics and dynamics in CAM.

2.2.1 Application Description

The Community Atmosphere Model provides the research community with a reliable, well documented atmospheric general circulation model. CAM has been developed over a period of fifteen years. It started as a community climate model that is a stand alone application and cannot be coupled with any other atmospheric or climate model. Over the years, CAM evolved into a more specific model of simulating and modeling the atmosphere. Also, the capability of being integrated into the Community Climate System Model (CCSM) was added. In CAM 3.0, many features and enhancements were added to it. The most important of which is the ability to support multiple dynamical cores instead of only one. In this work, Spectral Eulerian Dynamics is the core of focus.

The CAM 3.0 cleanly separates the parameterization suite from the dynamical core, and makes it easier to replace or modify each in isolation. The dynamical core can be coupled to the parameterization suite in a purely time split manner or in a purely process split one, as described below [1].

Consider the general prediction equation for a generic variable ψ

$$\frac{\partial\varphi}{\partial t} = D(\varphi) + P(\varphi) \tag{1}$$

where φ denotes a prognostic variable such as temperature or horizontal wind component. The dynamical core component is denoted ***D*** and the physical parameterization suite ***P***.

A three-time-level notation is employed which is appropriate for the semi-implicit Eulerian spectral transform dynamical core. However, the numerical characteristics of the physical parameterizations are more like those of diffusive processes rather than advective ones. They are therefore approximated with forward or backward differences, rather than centered three-time-level forms.

The *Process Split* coupling, which refers to the coupling of the dynamical core with the complete parameterization suite, is approximated by

$$\varphi^{n+1} = \varphi^{n-1} + 2\Delta t D(\varphi^{n+1}, \varphi^{n}, \varphi^{n-1}) + 2\Delta t P(\varphi^{*}, \varphi^{n-1}) \qquad (2)$$

where $P(\varphi^{*}, \varphi^{n-1})$ is calculated first from

$$\varphi^{*} = \varphi^{n-1} + 2\Delta t P(\varphi^{*}, \varphi^{n-1}) \qquad (3)$$

The *Process Split* form is convenient for spectral transform models.

The *Time Split* coupling, which also refers to the coupling of the dynamical core with the complete parameterization suite, is approximated by

$$\varphi^{n-1} + 2\Delta t D(\varphi^{n+1}, \varphi^{n}, \varphi^{n-1}) \qquad (4)$$

$$\varphi^{*} + 2\Delta t P(\varphi^{n+1}, \varphi^{*}). \qquad (5)$$

The *Time Split* form is convenient for the finite-volume core which adopts a Lagrangian vertical coordinate.

The distinction is that in the *Process Split* approximation the calculations of ***D*** and ***P*** are both based on the same past state, φ^{n-1}, while in the *Time Split* approximations ***D*** and ***P*** are calculated sequentially, each based on the state produced by the other.

As mentioned above, the Eulerian core employs the three-time-level notation in (Equation 2)-(Equation 5). (Equation 2)-(Equation 5) also apply to two-time-level semi-Lagrangian and finite volume cores by dropping centered n term dependencies, and replacing ***n***-1 by ***n*** and $2\Delta t$ by Δt.

The parameterization package can be applied to produce an updated field as indicated in (Equation 3) and (Equation 5). Thus (Equation 5) can be written with an operator notation

$$\varphi^{n+1} = P(\varphi^{*}) \qquad (6)$$

where only the past state is included in the operator dependency for notational convenience. The implicit predicted state dependency is understood. The *Process Split* equation (Equation 2) can also be written in operator notation as

$$\varphi^{n+1} = D(\varphi^{n-1}, \frac{P(\varphi^{n-1}) - \varphi^{n-1}}{2\Delta t}) \qquad (7)$$

where the first argument of ***D*** denotes the prognostic variable input to the dynamical core and the second denotes the forcing rate from the parameterization package, e.g. the heating rate in the thermodynamic equation. Again only the past state is included in the operator dependency, with the implicit predicted state dependency left understood. With this notation the *Time Split* system (Equation 5) and (Equation 5) can be written

$$\varphi^{n+1} = P(D(\varphi^{n-1}, 0)) \qquad (8)$$

The total parameterization package in CAM 3.0 consists of a sequence of components, indicated by

$$P = \{M, R, S, T\} \qquad (9)$$

where ***M*** denotes (Moist) precipitation processes, ***R*** denotes clouds and Radiation, ***S*** denotes the Surface model, and ***T*** denotes Turbulent mixing. Each of these in turn is subdivided into various components: ***M*** includes an optional dry adiabatic adjustment (normally applied only in the stratosphere), moist penetrative convection, shallow convection, and large-scale stable condensation; ***R*** first calculates the cloud parameterization followed by the radiation parameterization; ***S*** provides the surface fluxes obtained from land, ocean and sea ice models, or calculates them based on specified surface conditions such as sea surface temperatures and sea ice distribution. These surface fluxes provide lower flux boundary conditions for the turbulent mixing ***T*** which is comprised of the planetary boundary layer parameterization, vertical diffusion, and gravity wave drag [1].

Further details of the splitting of parameterized physics and the dynamical core can be found in [1]. Also, the detailed scientific explanation of the physics and dynamics involved in CAM can be found in [1] sections 2.2 and 3.2 respectively.

2.2.2 Control Flow

CAM, as mentioned earlier, can be divided into four major kernels in addition to INITIALIZATION and FINALIZATION. CAM starts execution in the cam subroutine in cam.F90 file. In this subroutine, all the initializations and finalization routines and calls take place. In this section, INITIALIZATION and FINALIZATION kernels will be discussed as the remaining kernels will be discussed in details in the following section. The main core of CAM execution is done in the stepon function in stepon.F90 file. This function has a time loop that calls the four major kernels.

During INITIALIZATION, dataset files are read and all the SPMD MPI communications are initialized. On the other hand, during FINALIZATION, history and restart files are written. It is obvious that these two kernels are heavily dependent on the I/O system and thus their performance and behavior are not completely predictable. Figure 1 and Figure 2 show CAM flow of execution.

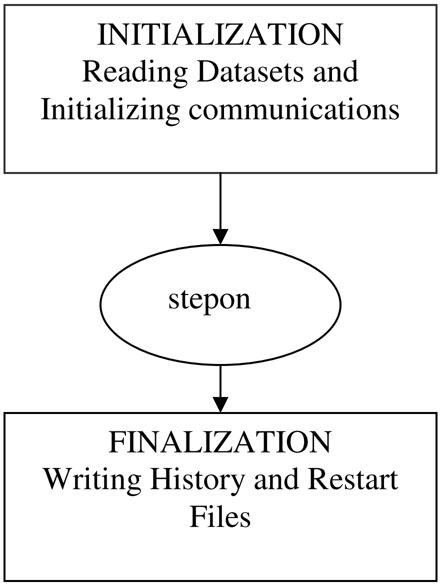

Figure 1: CAM Flow of Execution

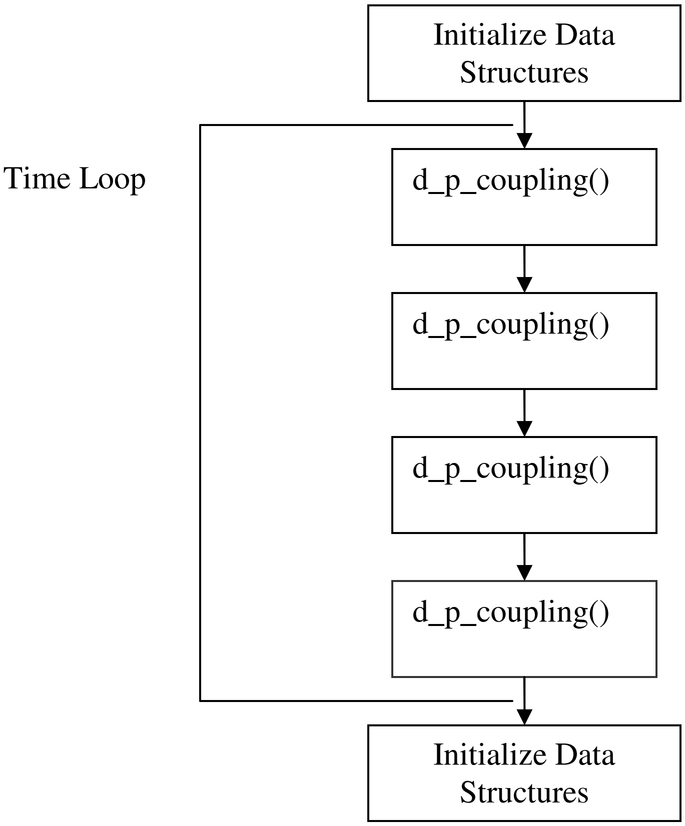

Figure 2: CAM Flow Inside Stepon

Table 2: T31 D_P_COUPLING Data Structure Sizes in Bytes

Name	Type and Description	2 Processors	4 Processors
ps	real(r8), intent(in) :: ps (plon, beglat:endlat)	12288	6144
t3	real(r8), intent(in) :: t3 (plon, plev, beglat:endlat)	319488	159744
u3	real(r8), intent(in) :: u3 (plon, plev, beglat:endlat)	319488	159744
v3	real(r8), intent(in) :: v3 (plon, plev, beglat:endlat)	319488	159744
q3	real(r8), intent(in) :: q3 (plon, plev, ppcnst, beglat:endlat)	958464	479232
omga	real(r8), intent(in) :: omga(plon, plev, beglat:endlat)	319488	159744
phis	real(r8), intent(in) :: phis(plon, beglat:endlat)	12288	6144
pdeld	real(r8), intent(in) :: pdeld (:,:,beglat:)		
phys_state	type(physics_state), intent(out), dimension(begchunk:endchunk) :: phys_state	10027008	5013504
phys_tend	type(physics_tend), intent(out), dimension(begchunk:endchunk) :: phys_tend	1327104	663552
pbuf	type(pbuf_fld), intent(inout), dimension(pbuf_size_max):: pbuf		

Table 3: T31 PHYS_PKG Data Structure Sizes in Bytes

Name	Type and Description	2 Processors	4 Processors
phys_state	type(physics_state), intent(inout), dimension(begchunk:endchunk) :: phys_state	10027008	5013504
phys_tend	type(physics_tend), intent(inout), dimension(begchunk:endchunk) :: phys_tend	1327104	663552
pbuf	type(pbuf_fld), intent(inout), dimension(pbuf_size_max) :: pbuf		

2.2.2.1 D_P_COUPLING

As mentioned previously, one of the main enhancements added to CAM 3.0 is the ability to support multiple dynamical cores. This enhancement required a full decoupling between the PHYS_PKG and the DYN_PKG. This decoupling was in data structures used and the parallelism techniques. Thus the need for this kernel (D_P_COUPLING) and the kernel explained in the next section (P_D_COUPLING). Table 2 has details of data structures that this kernel works on.

As the name implies, D_P_COUPLING is the kernel responsible for copying the data structures produced by the dynamical core into the data structures used by the physics package. Table 3 shows the data structures used by the physics package and Table 4 shows the data structures used by the dynamical core. This kernel is composed of many loops that have OpenMP support if the machine has support for threading. This nested loops copy the arrays and data structures produced by the dynamics into the phys_state structure of the physics package. Most of the loops have three dimensions due to the three dimensional nature of the grid data (Latitude x Longitude x Level). The latitude dimension is the dimension that is parallelized using MPI. To illustrate, the number of latitudes assigned to each processor is linear to the number of processors running CAM. As will be discussed in the PHYS_PKG section, in the physics package, data are represented differently (chunks and columns) to achieve maximum parallelization. Thus, this copying of data is also responsible for changing the array data structures from (Latitude x Longitude x Level) dimensions to (Chunks x Vertical x Columns) dimensions.

Table 4: T31 DYN_PKG Data Structure Sizes in Bytes

Name	Type and Description	2 Processors	4 Processors
adv_state	type(advection_state), intent(inout) :: adv_state		
t2	real(r8), intent(inout) :: t2(plon,plev,beglat:endlat)	319488	159744
fu	real(r8), intent(inout) :: fu(plon,plev,beglat:endlat)	319488	159744
fv	real(r8), intent(inout) :: fv(plon,plev,beglat:endlat)	319488	159744
			0
etamid	real(r8), intent(in) :: etamid(plev)	208	208
cwava	real(r8), intent(inout) :: cwava(plat)	768	768
detam	real(r8), intent(inout) :: detam(plev)	208	208
flx_net	real(r8), intent(in) :: flx_net(plon,beglat:endlat)	12288	6144
ztodt	real(r8), intent(in) :: ztodt	8	8
ps	real(r8), intent(in) :: ps (plon, beglat:endlat)	12288	6144
t3	real(r8), intent(in) :: t3 (plon, plev, beglat:endlat)	319488	159744
u3	real(r8), intent(in) :: u3 (plon, plev, beglat:endlat)	319488	159744
v3	real(r8), intent(in) :: v3 (plon, plev, beglat:endlat)	319488	159744
q3	real(r8), intent(in) :: q3 (plon, plev, ppcnst, beglat:endlat)	958464	479232
omga	real(r8), intent(in) :: omga(plon, plev, beglat:endlat)	319488	159744
phis	real(r8), intent(in) :: phis(plon, beglat:endlat)	12288	6144
pdeld	real(r8), intent(in) :: pdeld (:,:,beglat:)		

2.2.2.2 P_D_COUPLING

P_D_COUPLING is the kernel responsible for doing exactly the opposite of the previous kernel. A point worth mentioning is that each dynamical core has its own D_P_COUPLING and P_D_COUPLING functions. Hence, each dynamical core can easily interact with the physics package. In P_D_COUPLING all the data structures that are updated or changed by the PHYS_PKG are then copied into arrays that can be used by the dynamics package. As the case with D_P_COUPLING, the nested loops do the copying of data and changing the dimensions as mentioned in the previous section. Table 5 shows details of data structures that this kernel utilizes.

Table 5: T31 P_D_COUPLING Data Structure Sizes in Bytes

Name	Type and Description	2 Processors	4 Processors
phys_state	type(physics_state),intent(in), dimension(begchunk:endchunk) :: phys_state	10027008	5013504
phys_tend	type(physics_tend), intent(in), dimension(begchunk:endchunk) :: phys_tend	1327104	663552
			0
t2	real(r8), intent(out) :: t2(plon, plev, beglat:endlat)	319488	159744
fu	real(r8), intent(out) :: fu(plon, plev, beglat:endlat)	319488	159744
fv	real(r8), intent(out) :: fv(plon, plev, beglat:endlat)	319488	159744
flx_net	real(r8), intent(out) :: flx_net(plon,beglat:endlat)	12288	6144
qminus	real(r8), intent(out) :: qminus(plon, plev, pcnst, beglat:endlat)	958464	479232
qnats	real(r8), intent(out) :: qnats(plon, plev, ppcnst, beglat:endlat)	958464	479232

2.2.2.3 PHYS_PKG

The PHYS_PKG kernel is the most dominant kernel in CAM. The PHYS_PKG is responsible for all the physical parameterizations and uses the phys_state structure as the main data structure. This data structure is a large and many computations are done on that data structure that causes this kernel to dominate the execution. As mentioned earlier, CAM has a three dimensional grid structure (Latitude x Longitude x Vertical). Because computation in the physics is independent between vertical columns, the inner loop over longitude is vectorizable. Coarser grain parallelism is exploited in the outer loop over latitude, via either MPI or OpenMP [6]. Thus, the loops in the physics parameterization package looks like this:

```
do j=1,nlat
        do k=1,nver
                do i=1,nlon
                        (physical parameterizations)
        enddo
    enddo
enddo
```

As of CAM 3.0, the design of the loop structure in the physics parameterization package has changed. To exploit vectorization, which is important to both vector based architectures and cache-based processor architectures to exploit fine-grain parallelism for long-instruction-word architectures, the computation of multiple columns was bundled into chunks [6]. Thus the new array structure is (pcols, never, nchunks), and the new loop structure is:

```
do j=1,nchunks
    do k=1,nver
        do i=1,ncols(j)
            (physical parameterizations)
        enddo
    enddo
enddo
```

With this new design of arrays and loops, the inner loop is again vectorizable, and the outer loop is the MPI or OpenMP parallel direction. CAM is a Fortran code, so the inner loop also runs sequentially over contiguous memory locations. As the chunk size (pcols and ncols) decreases, the cache locality increases and the parallelism exploitable at the outer loop level increases. In contrast, as the chunk size increases, the vectorization opportunities increase [6]. Details of the new data structures and their sizes are presented in Table 3.

2.2.2.4 DYN_PKG

In this work, the focus is on the Spectral Eulerian Dynamical core. In general, the dynamical core is responsible for advancing the evolution equations for the atmospheric flow. The DYN_PKG is the second major kernel in CAM and the second dominant kernel in execution time. Details of data structures used by the DYN_PKG are presented in Table 4.

2.3 Kernel Coupling

The coupling parameter, C_{ij}, quantifies the interaction between adjacent kernels in an application [5]. In this work, four major kernels are identified, PHYS_PKG, DYN_PKG, P_D_COUPLING and D_P_COUPLING. A detailed explanation of each kernel will be provided in the CAM explanation section. 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 k*ernel 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 [8]. 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, it is computed as the ratio of the actual performance of the kernels together to that of no interaction, as given below:

$$C_{ij} = \frac{P_{ij}}{P_i + P_j}$$

For the case of a chain of kernels, **S** is defined as the set of kernels to be measured. The performance of the kernels is measured 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:

$$C_s = \frac{P_s}{\sum_{k \in S} P_k}$$

The parameters are grouped 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).

For example, for a chain of 3 (K1-K2-K3) in CAM the coupling value equation will be:

$$C_{12} = \frac{P_{123}}{P_1 + P_2 + P_3}$$

This coupling value can be used in predicting the performance of CAM. The equation used in performance prediction is:

$$T = P_{init} + \sum_{i=1}^{4} \alpha_i N_i P_i + P_{final}$$

where α_i is the weighted average of the kernel coupling values associated with kernel *i*. α_i can be calculated using:

$$\alpha_i = \frac{\sum_{j \in Q} c_j \times p_j}{\sum_{j \in Q} p_j}$$

Where *Q* is the set of all coupled kernels involved with kernel *i*.

In this work, kernel pairs and chains of three kernels were executed. Each kernel was run separately in a loop of 500 iterations. The choice of the number of 500 was to make sure that data starts to stabilize in cache and any other data from previous functions are out. The kernel pairs (D_P_COUPLING, PHYS_PKG), (PHYS_PKG, P_D_COUPLING), (P_D_COUPLING, DYN_PKG) and (DYN_PKG, D_P_COUPLING) were also executed in loops of 500 iterations. For some kernels, some tweaking was needed in order not to blow up the model and to keep the data within certain ranges that the model can tolerate. Finally, chains of three kernels were also run in loops of 500 iterations, (D_P_COUPLING, PHYS_PKG, P_D_COUPLING), (PHYS_PKG, P_D_COUPLING, DYN_PKG), (P_D_COUPLING, DYN_PKG, D_P_COUPLING) and (DYN_PKG, D_P_COUPLING, PHYS_PKG). Using the coupling values from these runs, an application model was generated. This application model was used in predicting CAM execution runtime and performance.

3. EXPERIMENTAL RESULTS

As mentioned previously, CAM was executed on four different parallel platforms in order to identify its general behavior and characteristics. The characteristics that were of concern for these tests were scalability, execution time and communication. In this section, a detailed analysis for these characteristics will be shown along with the execution results that indicate these characteristics.

3.1 Processor Partitioning

The aim of processor partitioning analysis is to identify the application factors that impact the selection of the best number of processors per node to use for execution of MPI applications. Thus, the focus of this analysis is the MPI-only version of CAM. The current trend in parallel systems is shifting towards clusters of shared memory symmetric multiprocessors (SMP), with moderate number of processors per node [7]. Hence, this analysis will identify the best configuration to run the MPI-only version of CAM and, also, will provide further insight on CAM characteristics and behavior.

To analyze the performance of CAM, it was executed on DataStar P655, P690, PSC Lemieux and NERSC Seaborg. The total number of processors was kept constant while changing the number of processors per node to see the effect of such configuration. The total runtime, communication time and initialization were collected in order to see the effect on both communication and computation. Initialization was an important factor due to its heavy reliance on I/O. Thus, initialization time needed to be calculated to be subtracted from total execution time to have accurate computation and communication timings. Tables 7, 8, 9 and 10 show the results of the tests of 32 processors on the four aforementioned platforms.

Table 6: Bi-directional Latency and Bandwidth Using *Sendrecv*

Platform	**Communication Mode**	**MPI Latency (µs)**	**MPI Bandwidth (MB/s)**
P655	Intra-node (1x2)	2.90	3724.01
	Inter-node (2x1)	6.71	1600.55
P690	Intra-node (1x2)	4.91	2606.86
	Inter-node (2x1)	8.01	1504.12
Seaborg	Intra-node (1x2)	14.45	932.84
	Inter-node (2x1)	29.89	295.61

**Lemieux data is not available*

Table 7: Processor Partitioning Data on Seaborg

T31 Resolution	**Runtime (secs)**	**Communication (secs)**	**Initialization (secs)**
2x16	51.414518	6.693528	10.0879
4x8	41.816107	4.723436	4.21962
8x4	42.012999	4.842055	4.68493
16x2	42.848741	5.116556	5.45826
32x1	45.17013	5.193585	6.121484
T42 Resolution			
2x16	79.074583	8.302817	11.372018
4x8	70.570393	4.258834	4.880712
8x4	71.811414	4.024824	6.311421
16x2	69.618445	4.257174	6.496892
32x1	71.742334	3.853449	7.93939
T85 Resolution			
2x16	395.91757	21.119348	31.804444
4x8	384.35363	16.084764	27.306053
8x4	381.27095	15.025156	27.12071
16x2	377.89918	15.151985	28.767705
32x1	370.620077	13.086333	24.424302

As it is indicated in [7], there are three major characteristics that affect the performance of a parallel application for the case when the number of requested processors is larger than the maximum number of processors per node. These characteristics are *Global Communication, Memory Access and Message Size*. It is obvious from the data shown in Tables 7, 8, 9 and 10 that CAM communication percentage of the total runtime is approx. 10% for T31 and approx. 5% for both T42 and T85. Thus, the interesting characteristic of CAM that this work focuses on is the memory access. The global communication, on the other hand, has very limited effect that only shows on the smallest dataset T31.

Each of the four platforms used shared a common factor. The four machines had a very high intra-node bandwidth (depending on shared memory), and lower inter-node bandwidth (using the underlying interconnection network). The intuition is that running the application with using the maximum number of processors per node will lead to the best performance. However, the aforementioned characteristics greatly affect the application performance. Inter-node and Intra-node bandwidth for each of the four platforms is shown in Table 6 [7].

For T31, the dataset size is the smallest. Thus, T31 is not as memory intensive as the rest of the datasets, and hence, less memory accesses. The lower the number of memory accesses the less memory congestion, hence the intra-node bandwidth is not totally consumed. This leads to a very consistent trend for T31 on P690, P655 and Lemieux where using the maximum number of processors per node yields the best performance. It is clear from the results that the communication time is shorter when using the maximum number of processors per node for these two machines. However, this is not the case for Seaborg. Seaborg has a more interesting outcome where the execution time starts being the longest for maximum number of processors per node which starts dropping by using less number of processors and then goes up again. The longer execution time experienced by the maximum number of processors per node is justified by memory congestion of 16 processors on the node and having less memory than P655 and P690. The increase in the execution time again when using less than half the processors on one node is justified by the slow interconnection network between nodes. This behavior is not encountered on DataStar due to the fast Federation Network used there.

Table 8: Processor Partitioning Data on P655

T31 Resolution	**Runtime (secs)**	**Communication (secs)**	**Initialization (secs)**
4x8	13.085524	1.182073	1.263196
8x4	16.022259	1.306746	3.970442
16x2	13.699135	1.537202	1.518381
32x1	13.675276	1.496823	1.374236
T42 Resolution			
4x8	24.230881	0.952962	1.602706
8x4	22.830676	0.950005	1.672912
16x2	22.656113	0.993427	1.846584
32x1	22.823563	1.030669	1.791539
T85 Resolution			
4x8	134.297046	4.932801	6.45849
8x4	125.44362	4.775869	6.407569
16x2	128.168296	4.025751	11.291061
32x1	122.027002	3.309613	7.550648

The T42 dataset experiences different behavior than that for the T31. This is due to the larger size of data of the T42 dataset. Since the sizes of the data structures are relatively larger than T31, memory congestion from array copying overhead is encountered. This congestion boosts the runtime. A point worth mentioning is that the communication time on P655 and Lemieux is still shorter for using max number of processors per node. This proves that communication overhead for CAM is relatively negligible to memory overhead. Nevertheless, P690 suffers from intra-node bandwidth consumption by memory congestion which leads to longer communication time for maximum number of processors per node. Also, Seaborg doesn't show the previous trend of the concave curve for neither runtime nor communication time. This happens when memory congestion starts to be the major overwhelming factor in the execution causing any network delay to be unnoticed.

Table 9: Processor Partitioning Data on P690

T31 Resolution	**Runtime (secs)**	**Communication (secs)**	**Initialization (secs)**
1x32	12.972556	0.966188	1.5539749
2x16	13.69839	0.992613	1.45267
4x8	13.69603	1.142692	1.482992
T42 Resolution			
1x32	27.33603	1.142692	1.7539749
2x16	24.705	0.992613	1.85267
4x8	22.589292	0.866188	1.782992
T85 Resolution			
1x32	130.648718	4.909863	7.881138
2x16	122.139668	2.994368	7.412044
4x8	112.778566	2.80783993	7.516255

Table 10: Processor Partitioning Data on Lemieux

T31 Resolution	**Runtime (secs)**	**Communication (secs)**	**Initialization (secs)**
8x4	38.013508	2.778082987	9.467733
16x2	34.22544	2.067937245	7.362273
32x1	34.018543	2.125226589	6.958006
T42 Resolution			
8x4	60.821712	2.7265488	11.469622
16x2	56.755615	2.783664	10.77925
32x1	59.077131	2.0358386	12.180661
T85 Resolution			
8x4	279.091735	7.3278648	37.307626
16x2	254.130367	6.48746	21.60738
32x1	248.882319	5.8346547	21.054645

The T85, as in Tables 7, 8, 9 and 10, with the largest dataset size shows yet another behavior where both communication time and computation time is the highest for maximum number of processors per node on all four platforms. As in the case of T42 on Seaborg where memory congestion is the controlling factor, memory congestion for T85 consumes all the intra-node bandwidth making even the intra-node communication slower than communication through the interconnection network. For T85 this trend is even experienced on P655 and Lemieux due to the huge data structures sizes.

CAM is very interesting in that the major performance difference occurs with between the scheme utilizing all the processors per node and half of the maximum number of processors per node, with half of the maximum number of processors per node being the better scheme. Further, there is very little difference in the execution time between using one to half of the maximum number of processors per node. When all the processors per node are used, congestion can occur due to data copies of arrays. When half of the maximum number of processors or fewer per node are used the intra-node bandwidth is sufficient [7].

3.2 Processor Scaling

In this section, runtime comparison, scalability and communication analysis is provided.

3.2.1 Execution Runtime Comparison

In the tests for runtime comparison among the four parallel platforms, CAM was configured to have one task per node. This configuration was necessary to guarantee to have one thread per processor and not to have multiple threads switching on the same processor. Each task had four OpenMP threads running on it. The choice of four threads was due to the fact that Lemieux has four processors per node and it was the least among the rest of the platforms. Thus the choice of four threads was to keep the workload consistent among the platforms and to have the workload per processor the same. This implies that on Lemieux, the maximum number of processors per node is used, while 50% of P655, and 25% of Seaborg capabilities per node is used. An exception from this configuration was DataStar P690. This exception is due to the fact that SDSC doesn't allow more than using four nodes on the P690. This limitation prevented running CAM on more than 128 processors on P690.

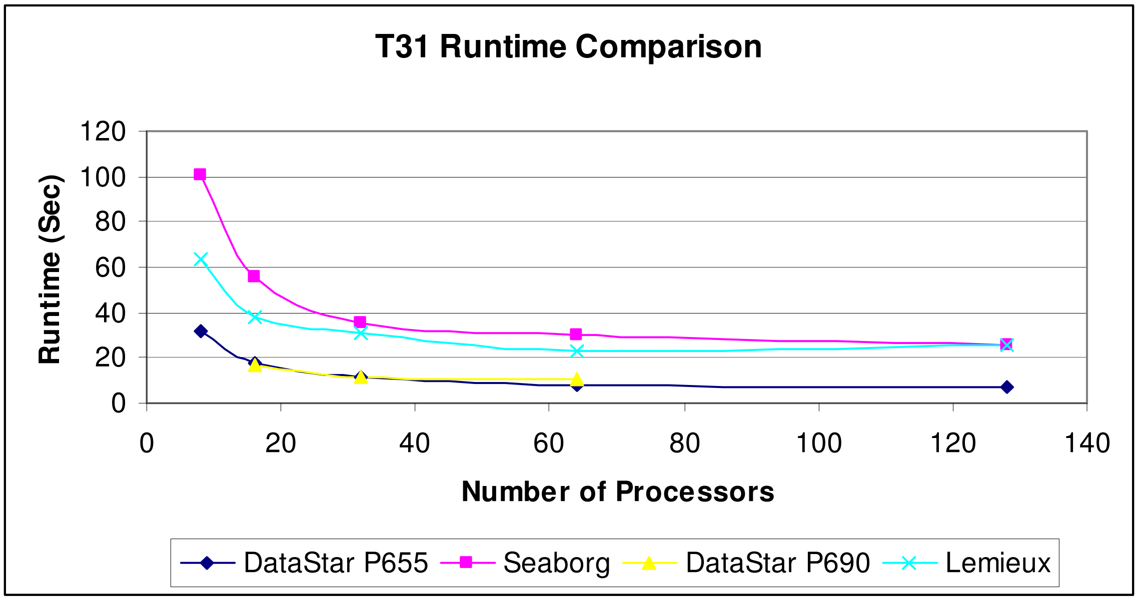

Figure 3: Runtime Comparison of the T31 Dataset on the Four Platforms

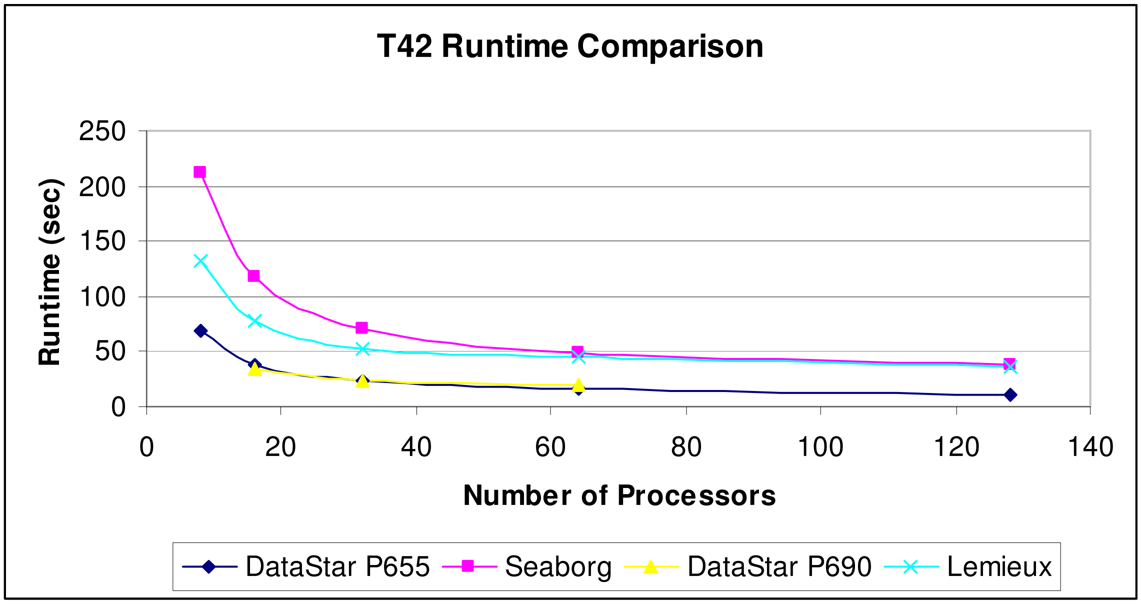

Figure 4: Runtime Comparison of the T42 Dataset on the Four Platforms

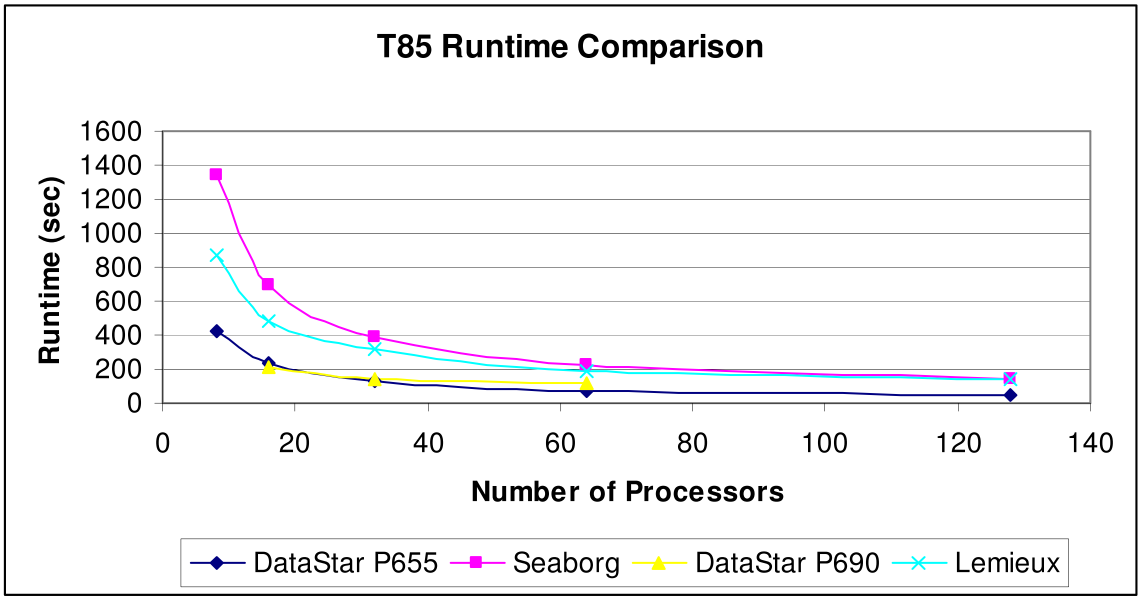

Figure 5: Runtime Comparison of the T85 Dataset on the Four Platforms

It is clear from Figures 3, 4 and 5 that runtimes don’t exactly reflect the platform architecture as expected. To illustrate, Seaborg with the slowest processors experience the longest runtime for all the input datasets; however, it is not four times slower that P655 as expected. This is due to the memory hierarchy of Seaborg and having a larger memory per processor and a larger L2 cache. This, in fact, shows the high dependability of CAM on memory.

Another point worth mentioning is the scalability on Lemieux. It is consistent among the three datasets that scalability on Lemieux is worse than the remaining machines. This scalability will be discussed in the next section.

3.2.2 Scalability

In the experiments for scalability, CAM was configured to run with number of OpenMP threads equal to the number of processors per node. This was the choice in order to have consistency among all machines and to avoid having the differences in scalability as indicated in the previous section. To illustrate, in the previous section, Lemieux runs were using the maximum processor capacity per node causing it to encounter the least scalability, while P655 was only using 50% of node capacity and

Seaborg was using 25% of load capacity. Thus the configuration for each machine was as follows:

- DataStar P655 : 1 Task/ Node – 8 Threads/ Task
- DataStar P690 : 1 Task/Node – 8 Threads/Task , 16 Threads/Task and 32 Threads/Task
- Seaborg : 1 Task/Node – 16 Threads/Task
- Lemieux : 1 Task/Node – 4 Threads/Task

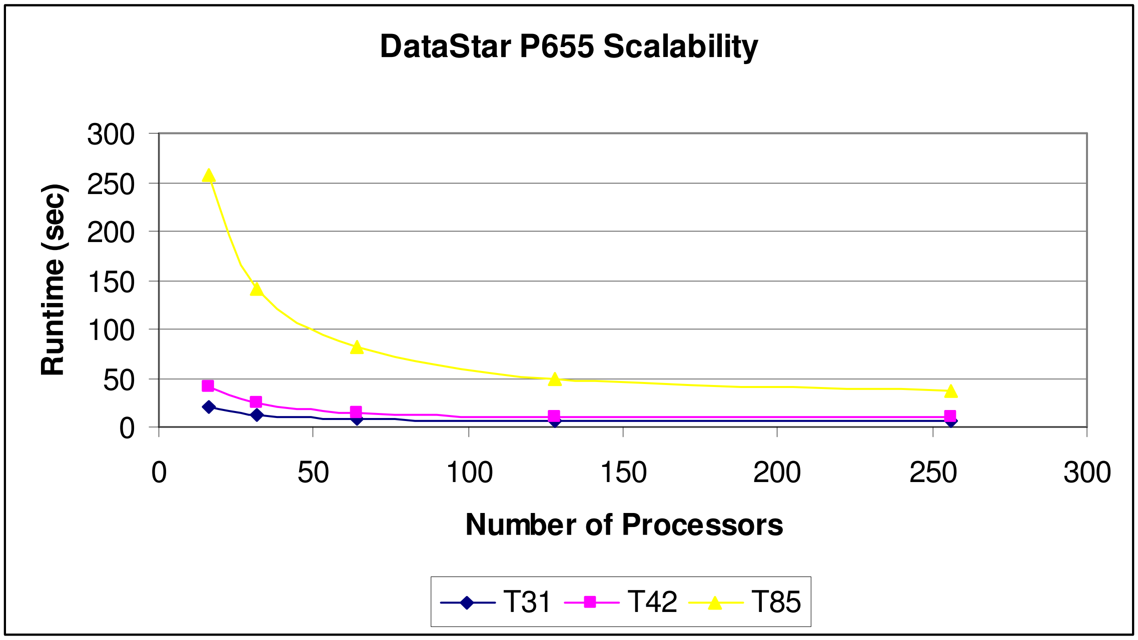

Figure 6: Scalability Comparison of the 3 Datasets on DataStar P655

Figures 6 and 7 show the runtime scalability of the three datasets on P655 and their relative speedup respectively. It is obvious from both graphs that CAM doesn't scale very well. This behavior is also consistent on all datasets and, as it is shown in Figures 8, 9, 10, 11, 12 and 13, it is also consistent on all the platforms.

In fact, this behavior is due to the intensive reliance of CAM on memory and the memory overhead incurred by the copying that occurs between different kernels and different data structures. This overhead is increased when all the processors on one node are used, thus using all the memory available on one node and memory thrashing occurs. Nevertheless, when half the number of processors per node is used or less, then there is enough memory per processor to accommodate the large data structures and the overhead of copying them.

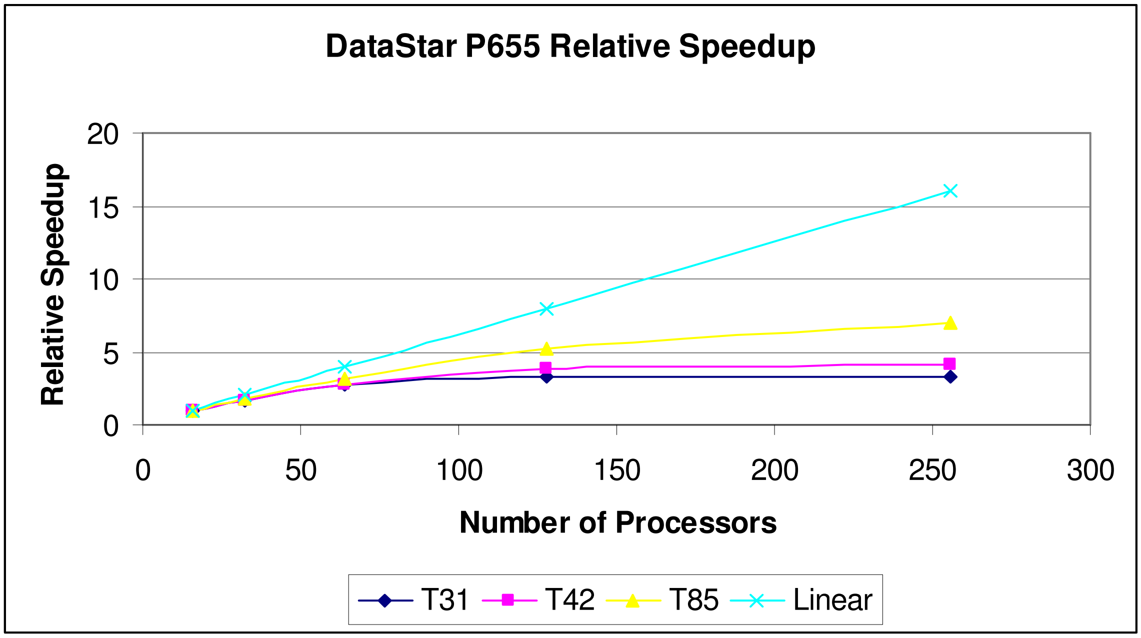

Figure 7: Relative Speedup of the 3 Datasets on DataStar P655

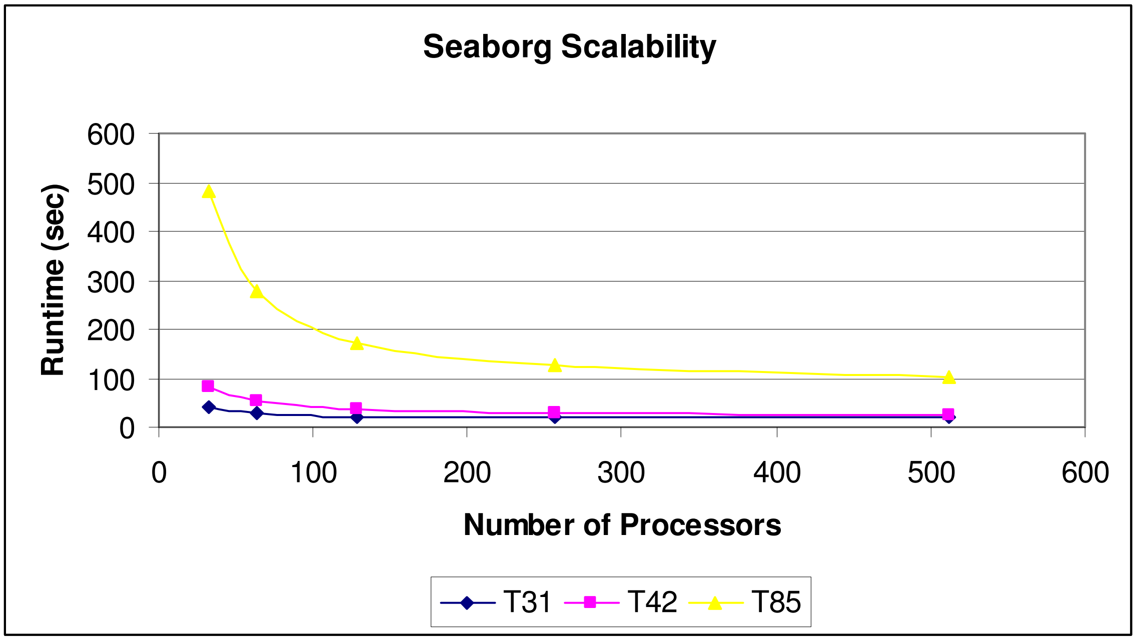

Figure 8: Scalability Comparison of the 3 Datasets on Seaborg

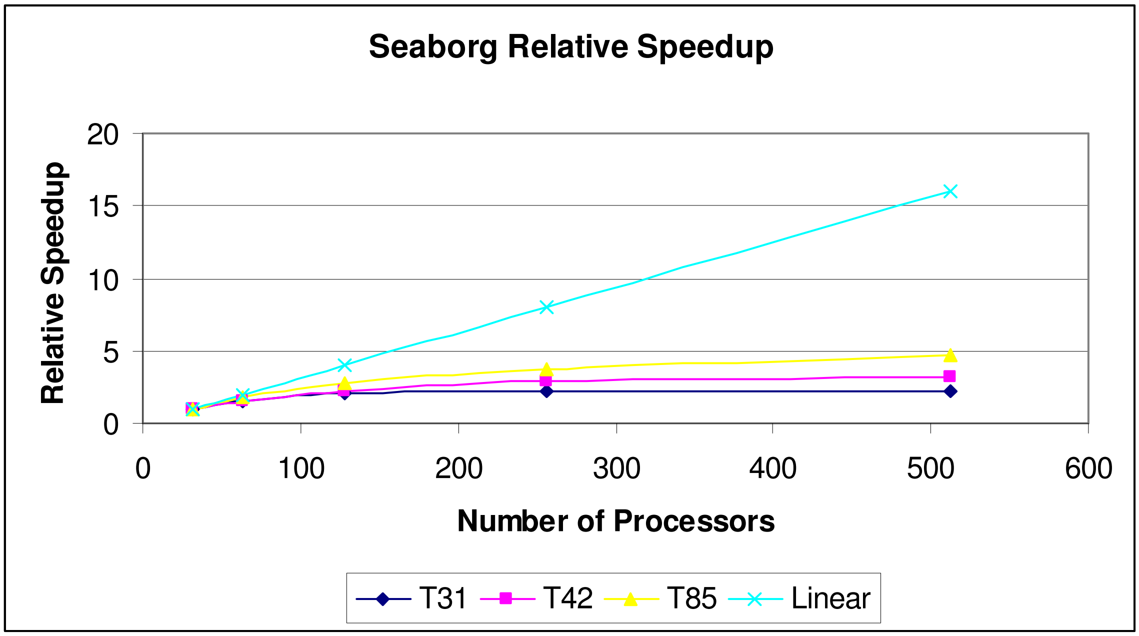

Figure 9: Relative Speedup Comparison of the 3 Datasets on Seaborg

Although, this may seem counter intuitive because it is known that the network bandwidth within the node is much larger than the inter-node network bandwidth, as it will be shown in the processor partition section, the communication overhead of CAM is much less than memory overhead. Also, the effect that communication has on CAM is considered negligible in comparison to the memory significant effect. Figures 14, 15 and 16 will show a comparison done on Seaborg where 16 threads per task run is compared to four threads per task run.

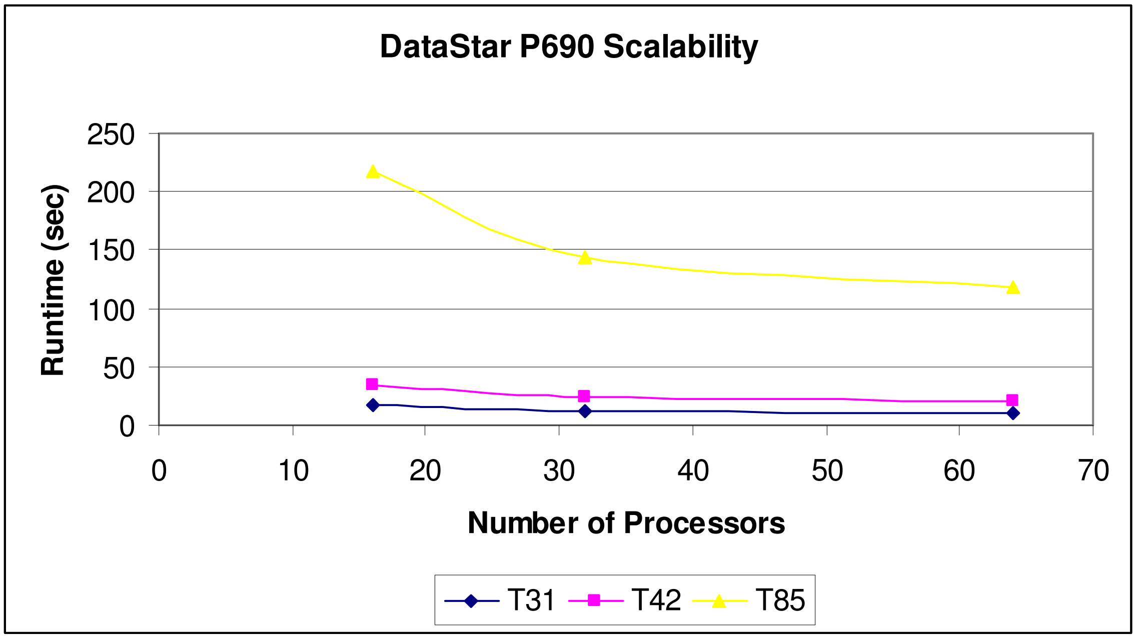

Figure 10: Scalability Comparison of the 3 Datasets on DataStar P690

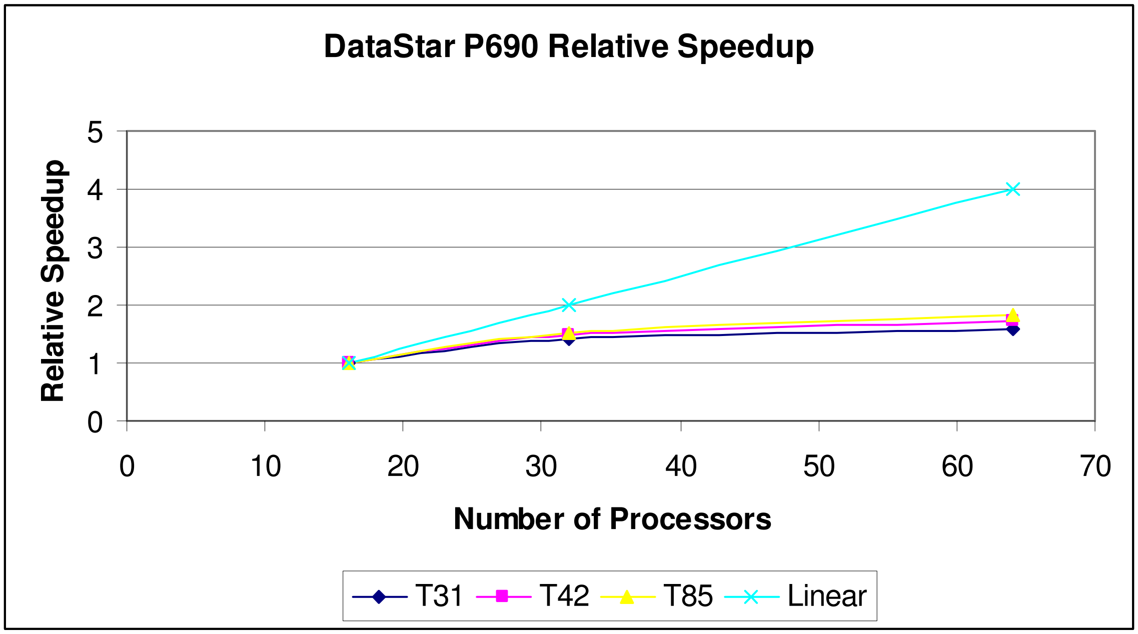

Figure 11: Relative Speedup Comparison of the 3 Datasets on DataStar P690

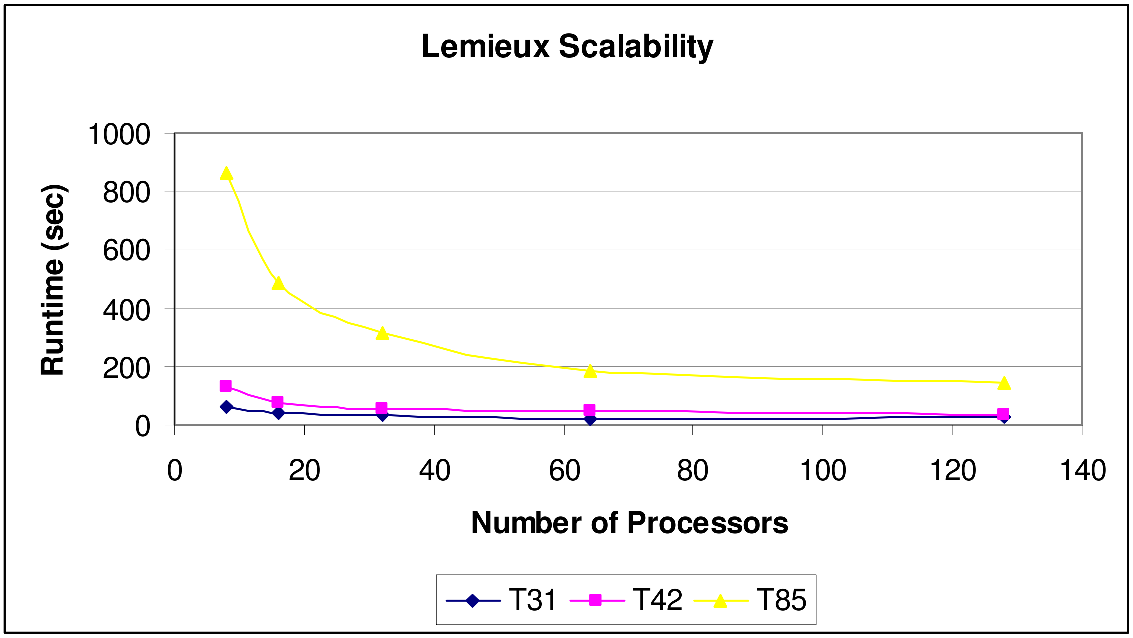

Figure 12: Scalability Comparison of the 3 Datasets on Lemieux

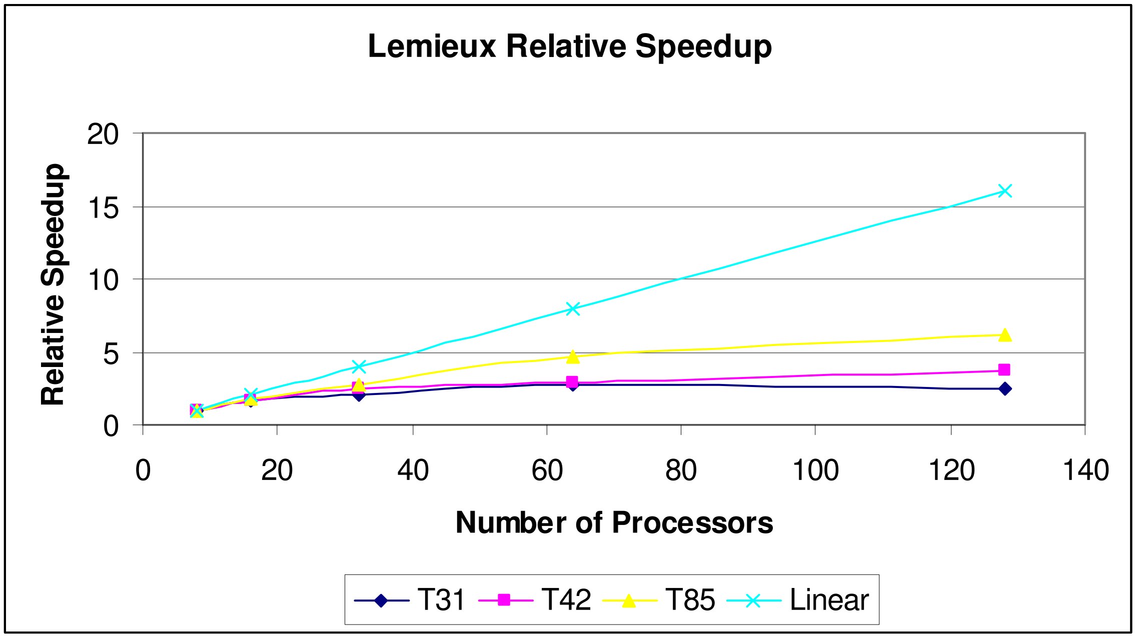

Figure 13: Relative Speedup Comparison of the 3 Datasets on Lemieux

Figures 14, 15 and 16 demonstrate the fact that using all the processors per node degrades the performance. For T31, for 32 processors the runtime is better if using 4 Threads; however, it degrades for larger number of processors. This is because for T31, data sizes and memory need, especially for larger number of processors, is not very demanding. Thus, communication overhead is the dominating factor. For T42, it can be easily seen that a similar behavior is encountered, although the effect is seen on a larger number of processors due to the bigger data sizes for the T42 dataset. On the other hand, the T85 behavior demonstrates the fact that using all the processors per node degrades the performance. Due to the large data sizes and the intensive demand for memory, communication overhead is negligible.

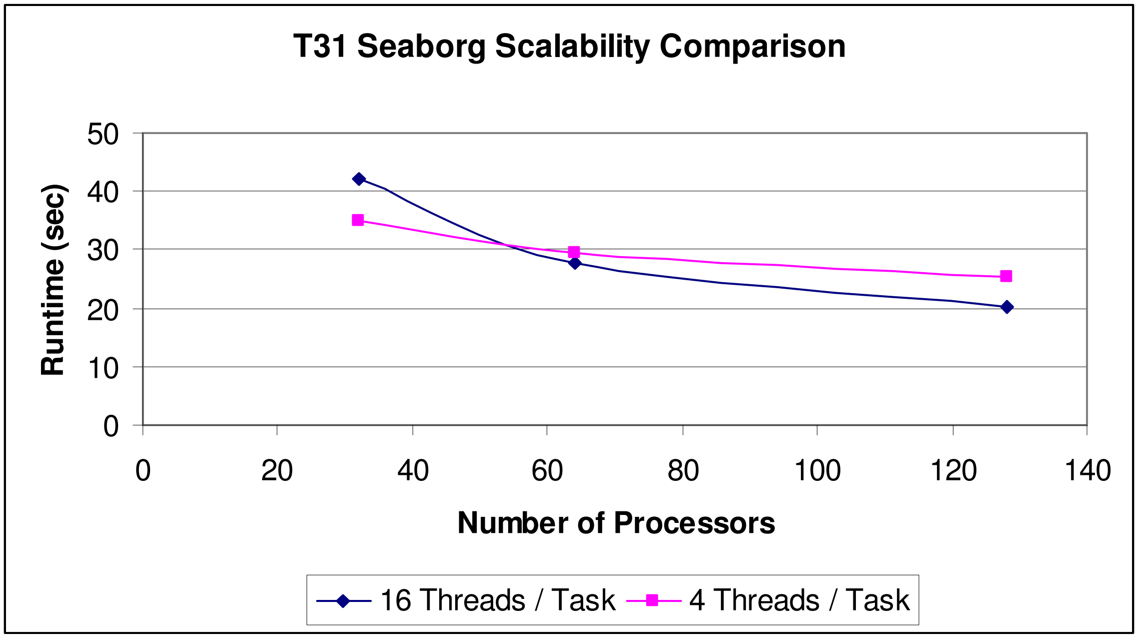

Figure 14: Scalability Comparison of the T31 Dataset on Seaborg with Different Number of Threads per Task

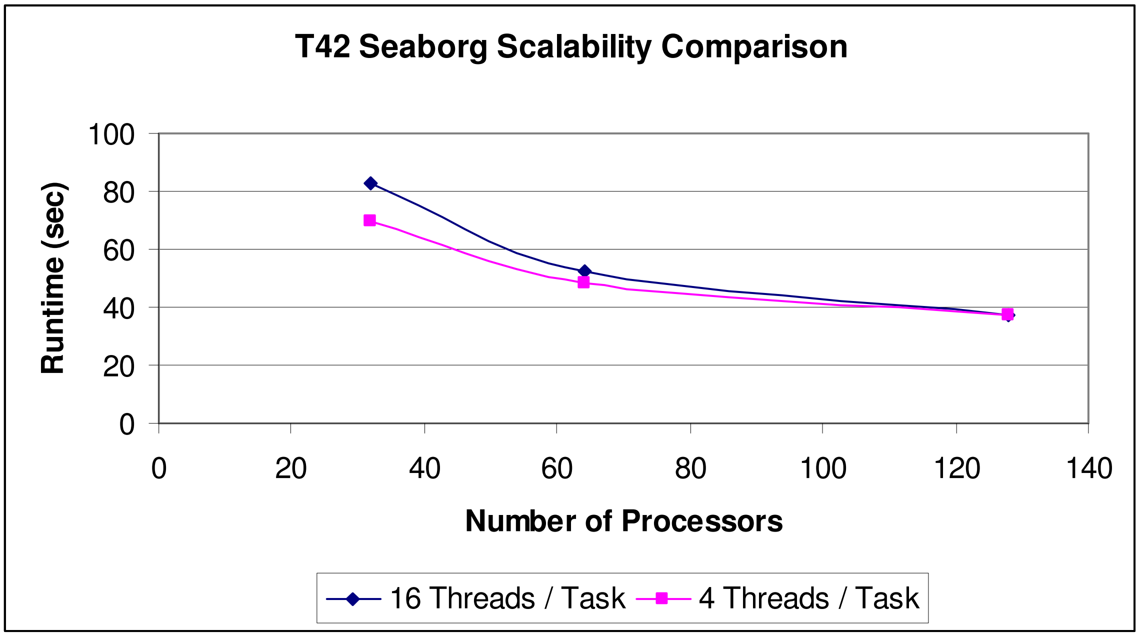

Figure 15: Scalability Comparison of the T42 Dataset on Seaborg with Different Number of Threads per Task

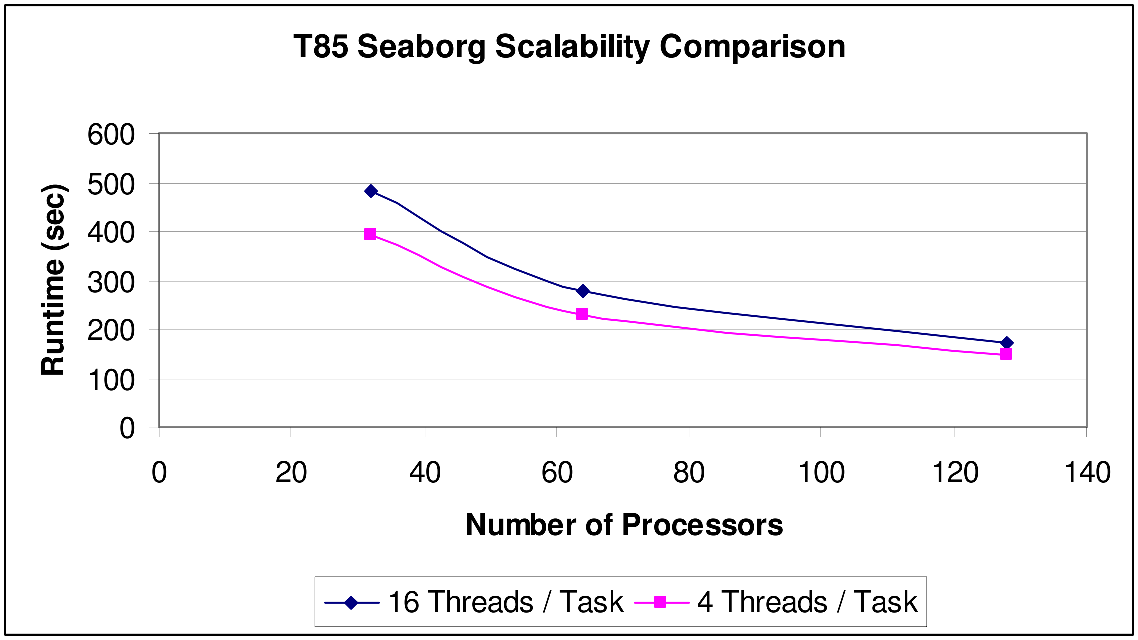

Figure 16: Scalability Comparison of the T85 Dataset on Seaborg with Different Number of Threads per Task

3.2.3 Communication

In the analysis of CAM's communication, MPI communication to computation ratio was measured for both the MPI-Only version and the HYBRID version. In both cases, MPI communication was obviously the same as the number of MPI tasks remains the same and hence the number of MPI calls remains the same. However, the ratio will normally change as the HYBRID version reaches higher processor count, therefore having less execution time. In all the tests for the communication, NERSC Seaborg and SDSC DataStar were the platforms of testing. In this section, the HYBRID model results will be shown and analyzed and in the processor partition section, the MPI only version will be analyzed in details.

The following two tables, Tables 11 and 12, show the execution time, MPI communication time on the master process, MPI communication time on non-master processes and the percentage of communication time of non-master processes to computation time. For all datasets and on both platforms, the trend is clear and stable. Communication time is relatively smaller than computation time, bearing in mind that in the HYBRID model there are more processors doing computation than the number of processors doing communication. To illustrate, in the case of 32 processors on Seaborg, only 2 processors will be responsible for the MPI communication, while the 32 processors will be doing computation. Thus, when we analyze the MPI only version in the processor partitioning section, it will be clear that communication is actually less than 5% of the total execution time. This, in fact, emphasizes the previous hypothesis that CAM is more memory intensive application than a communication intensive application.

Table 11: DataStar MPI Communication (seconds)

DataStar MPI Communication					
	Number of Processors				
	16	32	64	128	256
T31					
Actual Execution Time	19.40919	11.66729	7.147747	5.849899	5.806424
MPI Master Process Time	0.466595	0.479956	0.546961	0.586966	1.357719
MPI Time	3.640476	3.011145	2.843876	2.281166	2.86144
MPI Percentage	18.75646	25.80843	39.78703	38.99496	49.28059
Computation	15.76871	8.656149	4.303871	3.568733	2.944984
T42					
Actual Execution Time	40.62714	23.84297	14.87871	10.57473	9.932914
MPI Master Process Time	0.980445	0.947974	0.972482	1.019841	0.965429
MPI Time	4.010378	6.206893	5.92359	3.803363	3.7352
MPI Percentage	9.871181	26.03238	39.81253	35.96654	37.60427
Computation	36.61676	17.63608	8.955116	6.771364	6.197714
T85					
Actual Execution Time	258.0176	140.3475	81.87492	49.08211	36.65649
MPI Master Process Time	6.51962	5.037458	4.865098	4.33258	3.408755
MPI Time	21.54767	33.19722	31.66033	19.72617	14.41946
MPI Percentage	8.35124	23.65359	38.66914	40.19015	39.33671
Computation	236.4699	107.1503	50.21459	29.35594	22.23703

The trend that is encountered in this analysis is that with larger number of processors, communication time decreases but not in the same scale as computation time. The reason for that, as mentioned previously, is that number of processors doing computation is much more than those responsible for the MPI communication. Due to the less scalable communication time compared to computation time, the percentage of communication tends to be larger for larger number of processors.

Table 12: Seaborg MPI Communication (seconds)

Seaborg MPI Communication					
	Number of Processors				
	32	64	128	256	512
T31					
Actual Execution Time	44.22248	28.35822	20.64922	22.94409	21.9838
MPI Master Process Time	1.63369	2.113134	2.104961	2.305571	3.016865
MPI Time	7.652081	7.527077	6.268641	9.09134	9.004607
MPI Percentage	17.3036	26.54284	30.35776	39.62389	40.9602
	36.5704	20.83114	14.38058	13.85275	12.97919
T42					
Actual Execution Time	84.44865	54.10535	39.63757	29.7675	30.20784
MPI Master Process Time	3.774989	3.877889	3.670007	3.668254	3.827213
MPI Time	13.30371	13.51499	13.97302	10.34357	14.55347
MPI Percentage	15.75361	24.97903	35.25197	34.74786	48.1778
	71.14494	40.59036	25.66455	19.42393	15.65437
T85					
Actual Execution Time	482.8127	281.947	169.1986	128.1411	107.829
MPI Master Process Time	23.53569	20.48308	15.15449	13.92867	12.40494
MPI Time	57.99843	58.17592	45.96292	43.99574	39.81615
MPI Percentage	12.01262	20.63363	27.16506	34.33382	36.92528
	424.8142	223.7711	123.2357	84.14538	68.01281

3.3 Kernel Coupling

In this section, kernel coupling analysis is provided. A detailed analysis for each kernel pair is provided which can be extended to chains of three kernels.

3.3.1 Kernel Coupling Analysis

All the coupling values that were calculated for different kernel pairs or chains of three kernels were all very close to 1. The range of these coupling values was mostly between 0.9 and 1.1 with very few exceptions which will be explained and shown in this section. This trend of having the coupling values very close to 1 is due to the large data sizes of the data structures that each kernel use. Even for the smallest dataset, T31, data sizes are still large in comparison to the machines cache sizes. Thus, the data sharing and reuse between kernels is very limited.

3.3.1.1 K1-K2 Kernel Pair

K1 and K2 are the two kernels with the most data reuse as determined from Figure 20, Tables 2 and 3, and also as shown in Figures 17, 18 and 19. In all the K1-K2 graphs for all the datasets, the coupling is constructive in most case since the coupling values are between 0.9 and 1.0. This constructive coupling is due to the design of the data structures shared between these two kernels, corresponding to the phys_state array as well as the design of K1.

To further explain the reason why K1 and K2 have the most constructive coupling values among all kernels, a detailed explanation and analysis of K1 and its design is required. As shown in Figure 20, K1 execution is divided into two major sub-kernels. The first sub-kernel which consumes approximately 33% of the execution time of K1 on all machines -with the exception of Seaborg due to its 128-way set associative cache- is responsible for copying the dynamics' arrays into the arrays of the phys_state structure. The remainder of K1, approximately 67% of the execution time, is responsible for copying data within the phys_state structure itself to fill up the remainder of the structure.

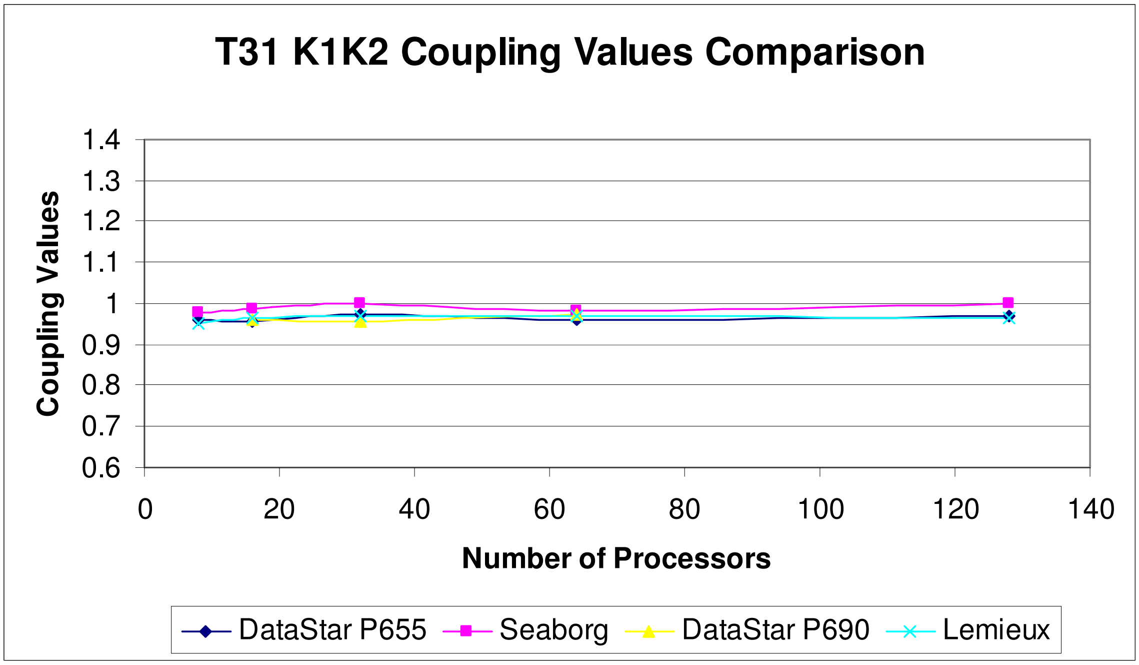

Figure 17: Coupling Values Comparison of the K1-K2 Kernel Pair for the T31 Dataset on the Four Platforms

Thus, as illustrated in Figure 20, for the last 67% of the execution of K1, phys_state is the only data that is being used by K1. Since K2 mainly uses the phys_state data structure, in addition to some local workspace variables, the coupling between the two kernels is constructive. Furthermore, by using a small number of columns per chunk in the phys_state structure, where a chunk is a collection of vertical columns of the grid, there is a great benefit by having high cache locality. Section 2.2.2.3 has further explanation on the chunk and columns data structures.

For the T31, T42 and T85 datasets, the coupling values tend to be close to one for all platforms, mostly ranging from 0.9 to 1.0. Also, the coupling values tend to be stable and equivalent for different number of processors. This is due to the nature of the phys_state array where number of columns per chunk is kept small to achieve high cache locality.

There is also another interesting fact that is clear in the graphs, coupling values don't decrease by increasing the number of processors and, also, they don't increase by increasing the size of data by using larger datasets. Going back to Tables 2 and 3, one would easily calculate the size of data per processor. For the T31, the smallest of all datasets, using 128 processors, there is 212736 Bytes per processor for data shared between kernels in addition to data structures that are local to each module or subroutine. This exceeds the size of D-Cache on all platforms. Furthermore, T42 data sizes are more than 200% the size of T31 data sizes and T85 exceeds 1000% the size of T31 data sizes. Thus, even with larger number of processors, the size of data is much bigger than the cache sizes. Hence, all the sharing that is encountered between K1 and K2 is mainly due to the cache locality of the phys_state and the design of K1 as previously mentioned.

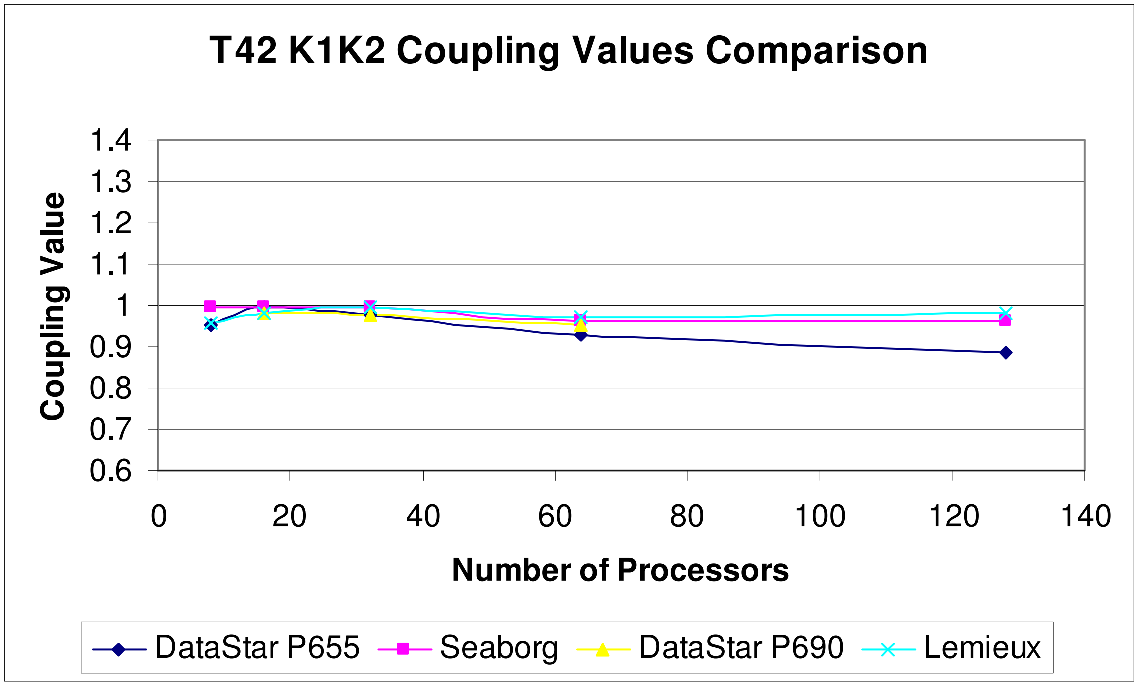

Figure 18: Coupling Values Comparison of the K1-K2 Kernel Pair for the T42 Dataset on the Four Platforms

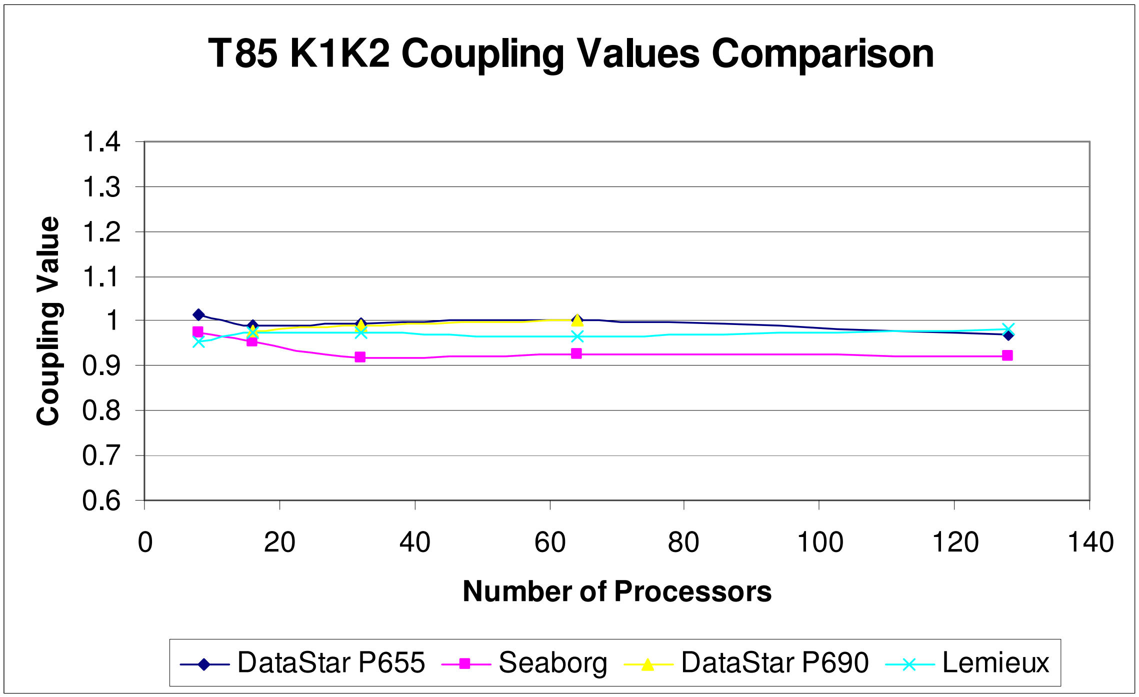

Figure 19: Coupling Values Comparison of the K1-K2 Kernel Pair for the T85 Dataset on the Four Platforms

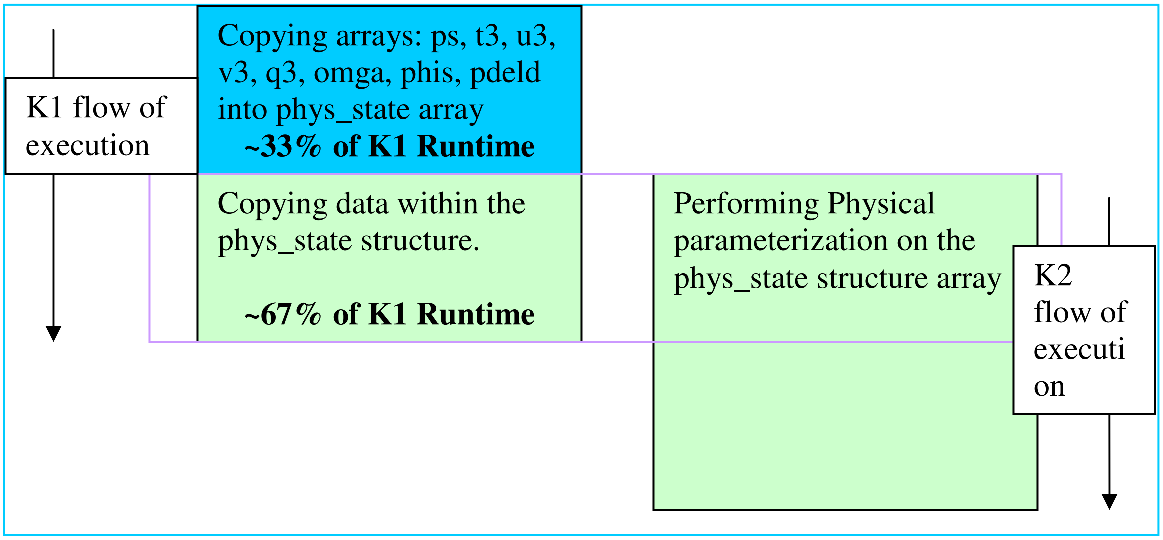
K1 flow of execution
Copying arrays: ps, t3, u3, v3, q3, omga, phis, pdeld into phys_state array
~33% of K1 Runtime
Copying data within the phys_state structure.
~67% of K1 Runtime
Performing Physical parameterization on the phys_state structure array
K2 flow of executi on

Figure 20: K1-K2 Kernel Pair Execution Illustration

3.3.1.2 K2-K3 Kernel Pair

K2-K3 kernel pair is the most interesting and most complicated kernel pair to analyze. As it is clear from the graphs in Figures 21, 22 and 23, K2-K3 has very high coupling values and hence experiencing destructive coupling. Since K2 is common on both K1-K2 and K2-K3 kernel pairs, but each kernel pair has different behavior, a comparison between K1 and K3 design, runtimes and trends on different machines need to be shown. In addition to having high coupling values, K2-K3 coupling values show high variation from one machine to another and from one dataset to another.

As shown in Figure 24, K2 and K3 don't experience the same trend as K1 and K2. K3 doesn't have two sub-kernels as in K1. K3 is only responsible for copying the phys_state data into dynamics arrays. Thus the phys_state array is not the only data structure residing in the caches when K2-K3 is executed as there is no overlapping period as in Figure 20.

To further analyze and understand the reasons why K2-K3 behavior is not as K1-K2, comparison between the runtimes of both and their trends on different machines is required. Table 13 shows the trends of the runtime of each of K1 and K3 on the different platforms. The main trend of focus is the runtime and which kernel is taking longer on which machine. In Table 13, an ↑ indicates longer execution time. To Illustrate, the first column of the table indicates that for the T31 dataset on DataStar (both p655, p690), K1 takes longer time to execute than K3.

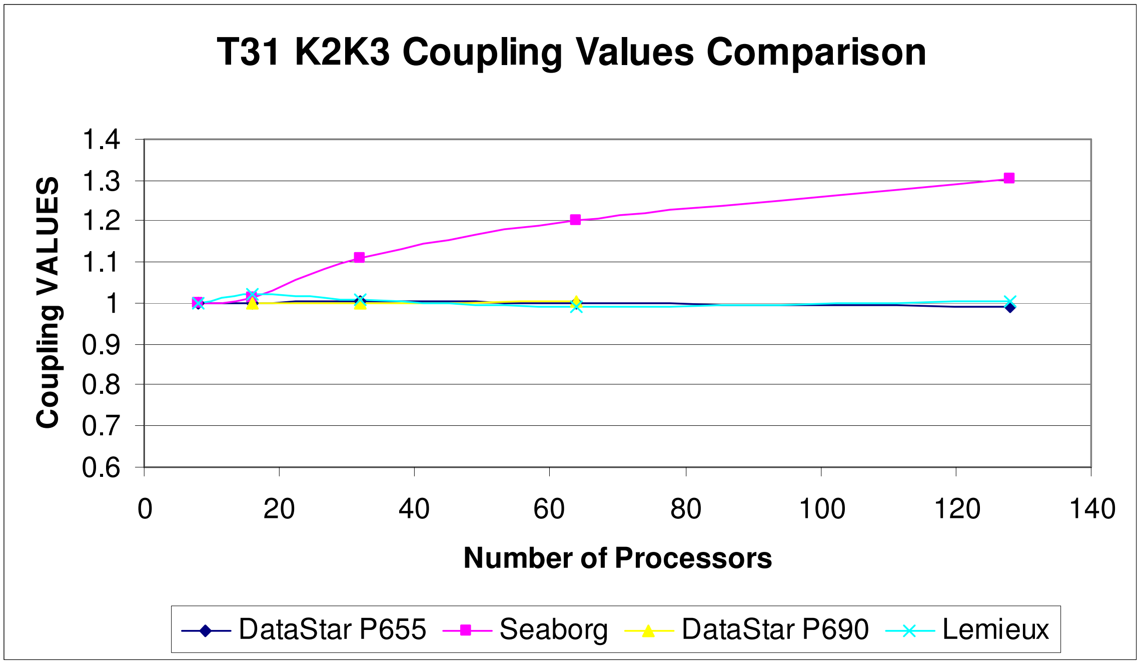

Figure 21: Coupling Values Comparison of the K2-K3 Kernel Pair for the T31 Dataset on the Four Platforms

Table 13: K1 and K3 Behavior on the Different Platforms

	T31			T42			T85		
Machines	DataStar	Seaborg	Lemieux	D	S	L	D	S	L
K1	↑	↓	↑	↑	↓	↑	↑	↑	↑
K3	↓	↑	↓	↓	↑	↓	↓	↓	↓

As it is clear from the table, Seaborg is the only machine with the odd behavior than all other platforms especially for the T42 and the T85 datasets. As it was mentioned earlier, Seaborg was the only exception in K1 sub-kernel runtime distribution. To illustrate, on DataStar and Lemieux, 33% of the execution of K1 was the copying of the dynamics arrays into the phys_state structure while 67% is copying data within the phys_state structure. With Seaborg the case is different. Approximately 49% of the execution time of K1 is in the first sub-kernel while only 51% is spent in the second sub-kernel. That implies, W.L.O.G that the second sub-kernel is executing faster on Seaborg. This is because the second sub-kernel is using only the same data structure which is characterized with high locality. Since Seaborg D-Cache is 128-way set associative cache, the hit rate for such data structure can be very high. Utilizing the Hardware Performance

Monitor utility on IBM machines, this hypothesis was shown correct. Seaborg L1 D-Cache hit rate was above 99% while DataStar hit rate was little below 84%. Thus on Seaborg, this high locality for K1 forces its execution time to be less than that for K3 where less locality is encountered. On the other hand, DataStar as well as Lemieux, with 2-way direct mapped D-Cache, more cache replacements will be experienced where phys_state data will keep thrashing in and out of cache, especially with CAM's large data sizes, and hence boosting K1 runtime, especially the second sub-kernel.

With the previous comparison between K1 and K3 and the comparison between the different platforms, the analysis for K2-K3 kernel turns to be straight forward. For the T31 case in Figure 21, Seaborg has the highest coupling values. By looking at Figure 25, the illustration of the case of Seaborg is easy. Since Seaborg has very high L1 hit rate and it tends to be biased towards data that has high locality, K3 execution time when run in isolation will tend to be lower than when K2-K3 pair is run. K2-K3 execution time will be boosted up since the locality achieved when running K3 in isolation is no longer achievable. This is clear in Figure 25. By applying the kernel coupling formula:

$$C_{23} = K_2K_3/K_2+K_3 \qquad (10)$$

where K2-K3 represents the runtime for running the kernel pair K2-K3, while K2 + K3 represents the sum of runtime for running K2 in isolation and running K3 in isolation.

Since K3 execution time will decrease when run in isolation, coupling values will be boosted. Also, by increasing the number of processors, the locality of K2-K3 will decrease. This was tested using IBM HPM and such results of K2-K3 are shown in Table 14 where higher average number of loads per TLB miss indicates higher locality. Nevertheless, with increasing data sizes, the locality achieved by running K3 in isolation on Seaborg will not be as beneficial as before. Figure 25 also shows this situation where large data sizes cause more data to be replaced from cache. This makes the K3 runtime to increase even when run in isolation. By going back to the equation 10 with higher value to K3, the coupling value will start approaching 1 again. Hence the graph in Figure 22 and Figure 23 show that coupling values on Seaborg are more stable than that for T31.

Table 14.A: T31 HPM Data

T31	2x1		4x1		8x1		16x1	
	Seaborg	P655	Seaborg	P655	Seaborg	P655	Seaborg	P655
% TLB misses per cycle	0.025	0.004	0.025	0.043	0.027	0.005	0.026	0.006
Avg number of loads per TLB miss	1152.183	5016.241	1152.32	4012.766	1104.085	3517.701	1131.362	3129.98
Total L2 data cache accesses	1.41	21108.76	1.425	20335.9	1.438	21704.36	1.476	23415.48
% accesses from L2 per cycle	0.252	4.148	0.247	3.849	0.235	3.913	0.215	3.864

B: T42 HPM Data

T42								
% TLB misses per cycle	0.032	0.003	0.032	0.004	0.033	0.472	0.031	0.022
Avg number of loads per TLB miss	889.459	5488.157	891.848	5030.967	889.99	3600.15	915.772	834.035
Total L2 data cache accesses	1.472	92996.01	1.493	92027.04	1.502	85872.47	1.561	90174.65
% accesses from L2 per cycle	0.253	4.295	0.252	4.228	0.241	3.835	0.223	3.892

C: T85 HPM Data

T85								
% TLB misses per cycle	0.028	0.004	0.027	0.005	0.025	0.004	0.026	0.059
Avg number of loads per TLB miss	1033.843	3940.349	1055.276	3210.11	1147.909	4096.679	1124.563	307.302
Total L2 data cache accesses	1.441	147891.5	1.439	147108.3	1.447	146992.4	1.484	145124
% accesses from L2 per cycle	0.25	4.042	0.245	4.105	0.234	4.107	0.215	3.98

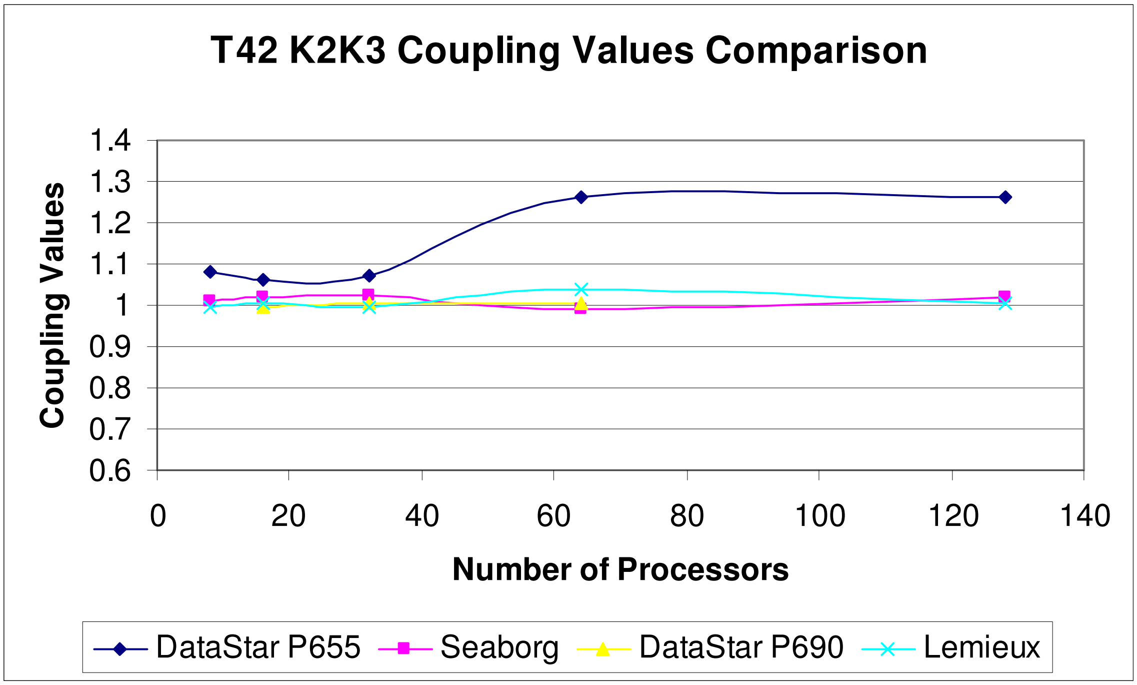

Figure 22: Coupling Values Comparison of the K2-K3 Kernel Pair for the T42 Dataset on the Four Platforms

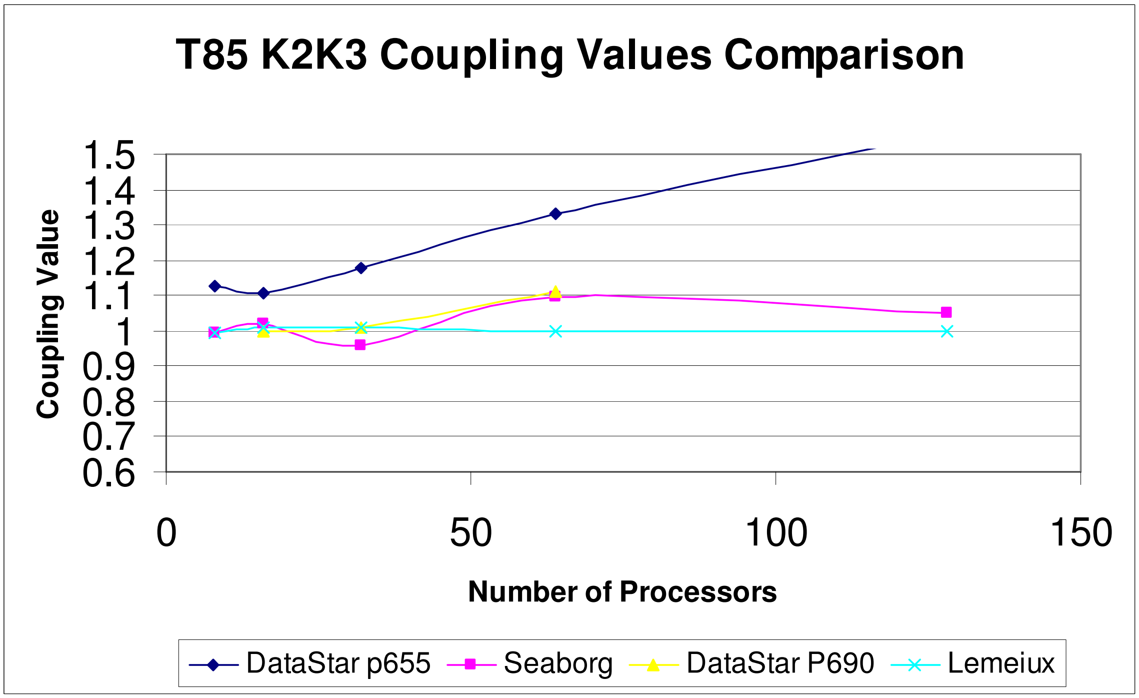

Figure 23: Coupling Values Comparison of the K2-K3 Kernel Pair for the T85 Dataset on the Four Platforms

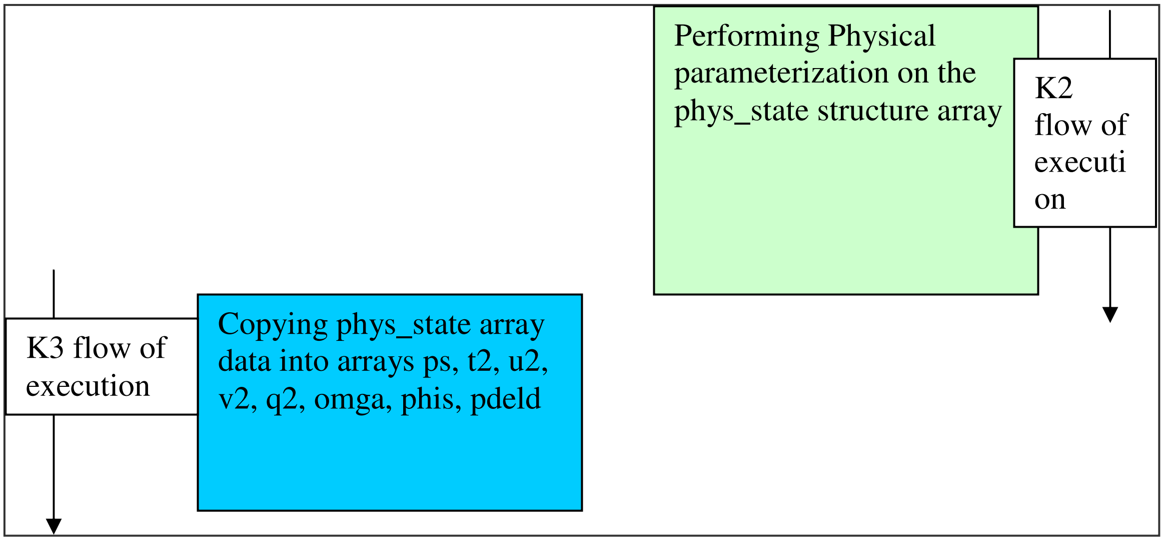
Performing Physical parameterization on the phys_state structure array
K2 flow of executi on
K3 flow of execution
Copying phys_state array data into arrays ps, t2, u2, v2, q2, omga, phis, pdeld

Figure 24: K2-K3 Kernel Pair Execution Illustration

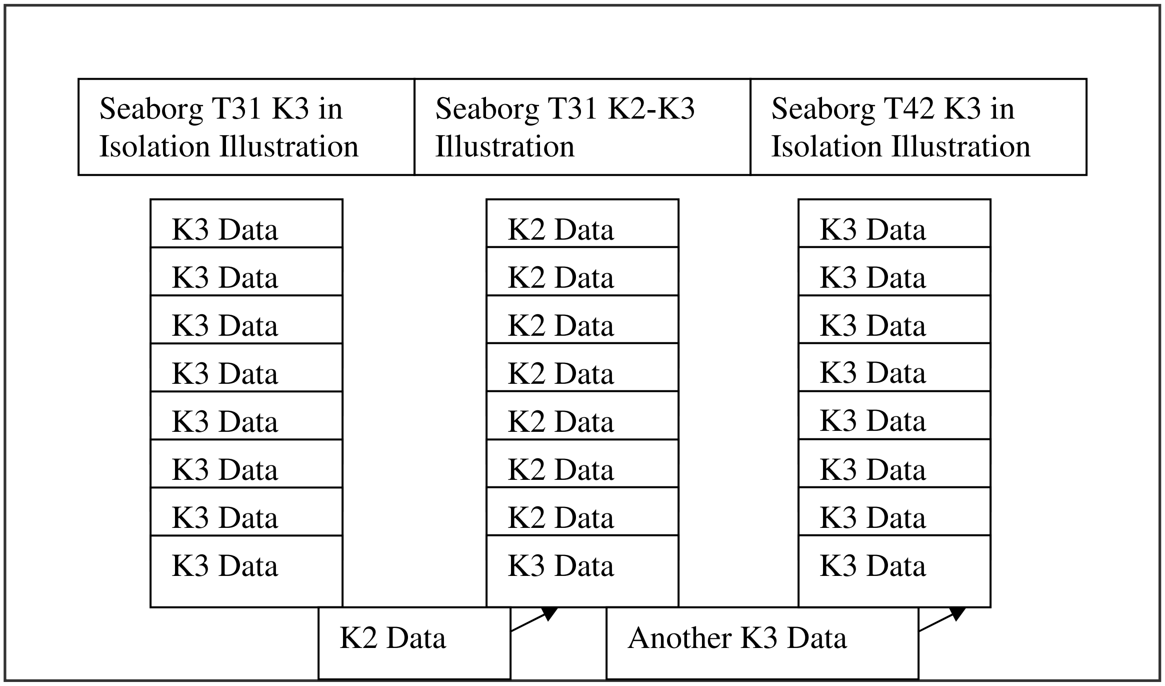

Figure 25: Seaborg Cache Behavior

As for DataStar, the same analysis for Seaborg applies. In the case of T31, Figure 26 shows how the 2-way direct mapped cache will behave when K3 is run in isolation. It is different from the case of Seaborg because the direct mapped cache doesn't make use of locality as much as Seaborg. Thus K3 runtime when run in isolation is still relatively close to the runtime of K3 when run within the kernel pair. Again applying that to equation 10, coupling values will be closer to one. Once again, with larger data sizes as in

T42 or T85, the situation where K2-K3 runtime starts to increase at a higher rate than K3 due to more thrashing of data causes the coupling values from equation 10 to increase. Also, by increasing the number of processors, HPM shows that locality decreases, causing K3 runtime when run in K2-K3 pair to increase even more with higher number of processors relative to K3 in isolation where more cache locality can still be achieved.

The case of Lemieux is unique. Lemieux follows DataStar Power4 2-way direct mapped cache policy; however, Lemieux has larger L1 cache size. This larger cache masks most of the effects of the locality and the replacement policies. It is easily noticed in all graphs of K1-K2 and K2-K3 that Lemieux coupling values are very stable and consistent. In the case of K1-K2, Lemieux has constructive coupling with very consistent values on all datasets. In the case of the K2-K3, coupling values are either 1 or little above 1. That shows that the larger L1 cache size is the key to better coupling.

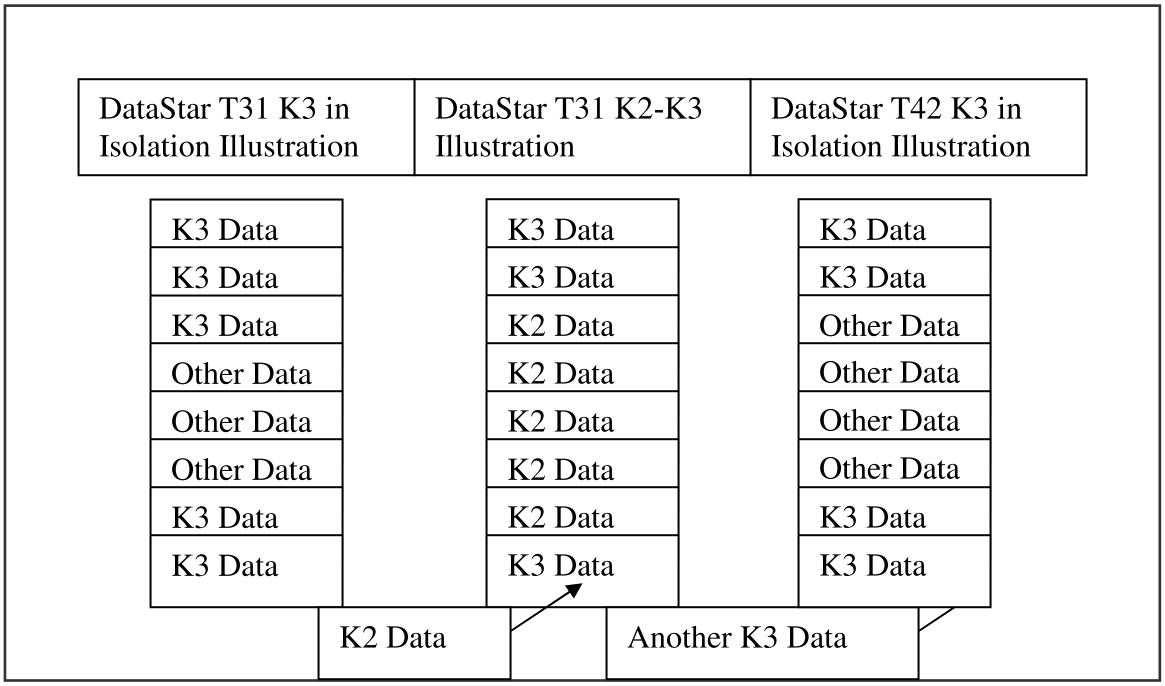

Figure 26: DataStar P655 Cache Behavior

3.3.1.3 K3-K4 Kernel Pair

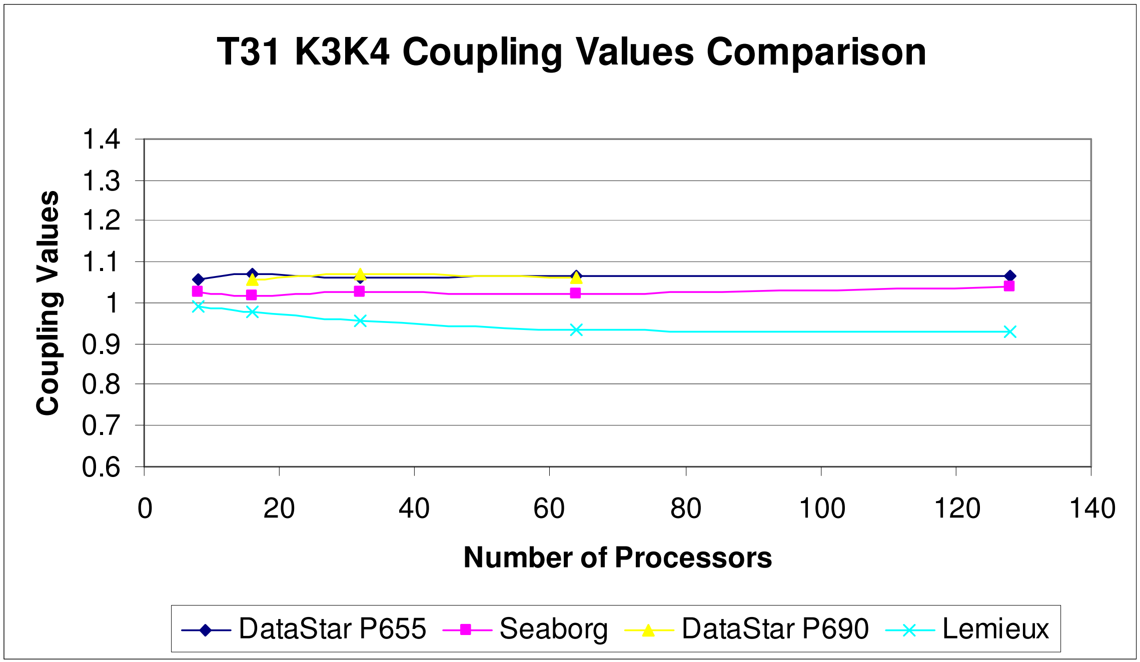

Figure 27: Coupling Values Comparison of the K3-K4 Kernel Pair for the T31 Dataset on the Four Platforms

K3-K4 kernel pair shows a consistent trend on all machines as shown in Figures 27, 28 and 29. The coupling values range from 0.9 to 1.1. An illustration of the interaction between K3 and K4 is in Figure 30. Lemieux is the only machine having coupling values below 1 for all datasets on all machines. As for the rest of the machines, coupling values are all above 1 but with a very little margin. K3 is the kernel responsible for copying the phys_state data into dynamics' arrays. These arrays account for approximately 40% of the data used in K4 as indicated in Table 5 and accounts for approximately 40% of the data used in K3 as indicated in Table 4. Although there may seem to be some sharing between the two kernels, the large sizes of phys_state data and the large sizes of dynamics' arrays, both accounting for more than 200Kbytes per processor for the T31 on 128 Processors, in K3 makes for a high cache miss rate on these data structures when running K3 in isolation or when run in K3-K4 kernel pair. Nevertheless, when run in K3-K4 kernel pair, the miss rate increases due to introducing the extra data structures in K4 as indicated in Table 5. This increase in cache miss rate accounts for the coupling values being slightly higher than 1. As for Lemieux, the larger cache size causes this miss rate to decrease and hence less coupling values.

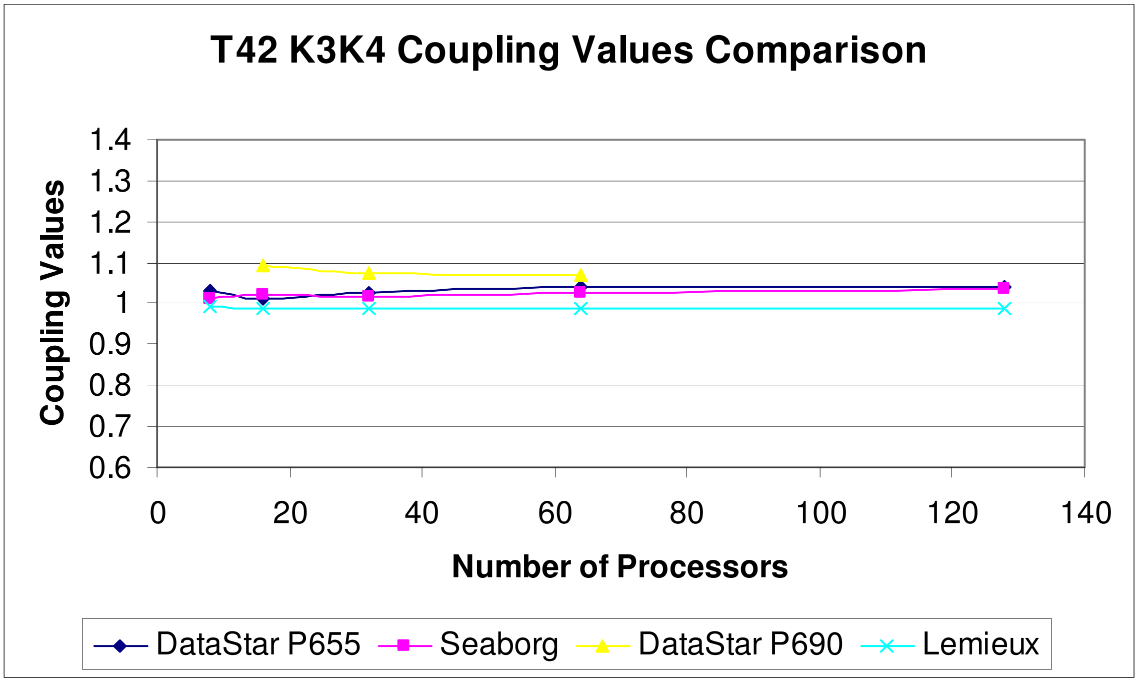

Figure 28: Coupling Values Comparison of the K3-K4 Kernel Pair for the T42 Dataset on the Four Platforms

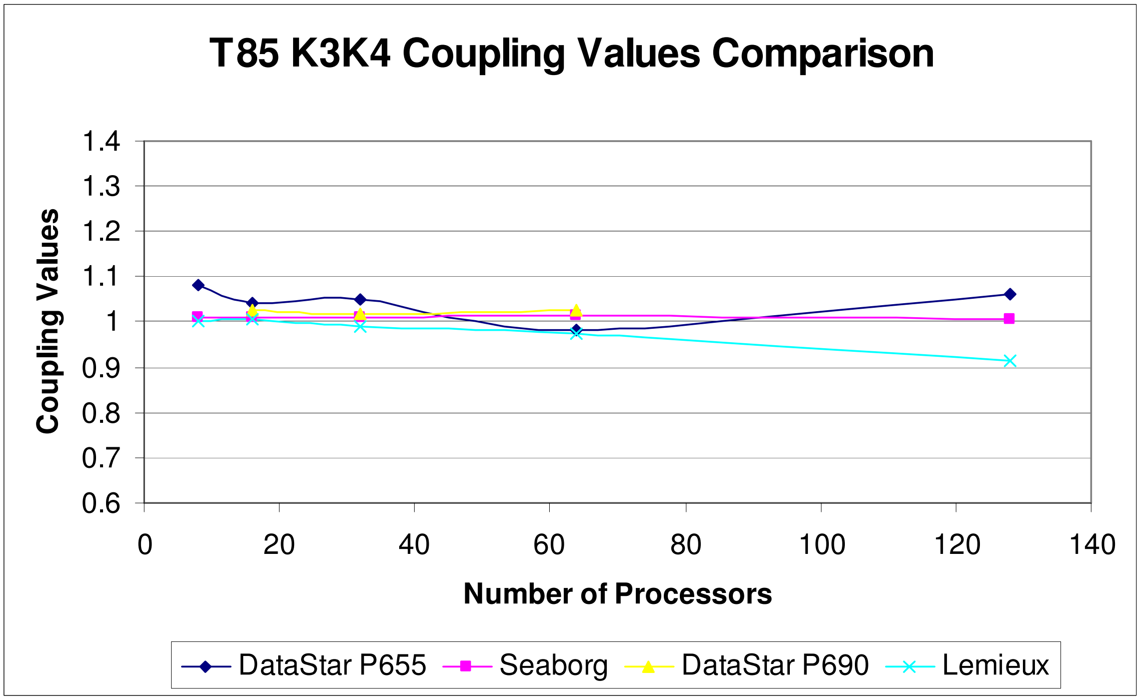

Figure 29: Coupling Values Comparison of the K3-K4 Kernel Pair for the T85 Dataset on the Four Platforms

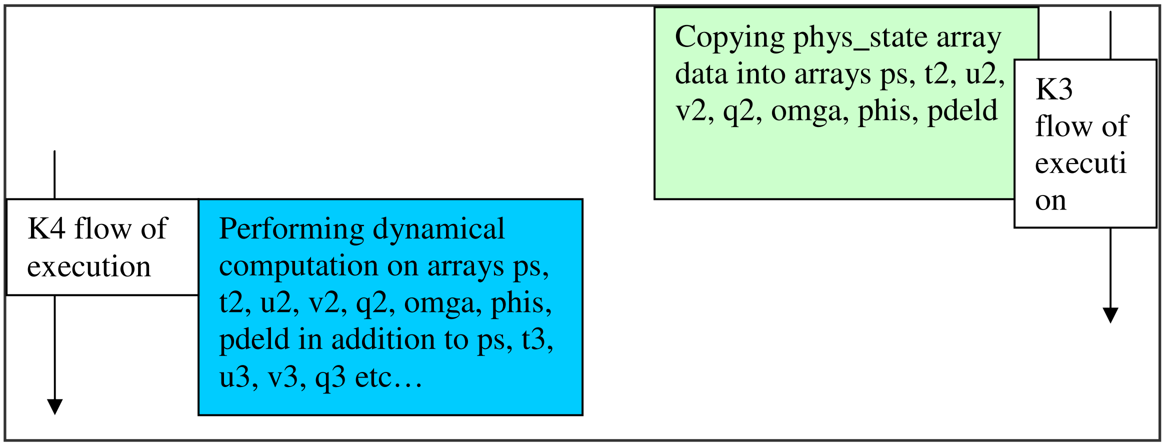
Copying phys_state array data into arrays ps, t2, u2, v2, q2, omga, phis, pdeld
K3 flow of executi on
K4 flow of execution
Performing dynamical computation on arrays ps, t2, u2, v2, q2, omga, phis, pdeld in addition to ps, t3, u3, v3, q3 etc…

Figure 30: K3-K4 Kernel Pair Execution Illustration

3.3.1.4 K4-K1 Kernel Pair

K4-K1 kernel pair is similar to K3-K4 kernel pair. K1 uses the arrays produced by K4 to copy them into the phys_state data structure. These arrays are ps, t3, u3, v3, q3, omga, phis and pdeld accounting for 40% of the data structure used in K1 and approximately 60% of the data structures used in K4. However, the use of the phys_state data structure accounting for 60% of the data used in K1 limits the sharing of the data between both kernels. The graphs for all datasets, Figures 31, 32 and 33, show consistent trend for the coupling values being all very close to 1.

On T31, Lemieux is the only machine having constructive coupling. This is due to the larger L1 cache and also having the largest L2 cache. The reader may argue that Seaborg has the same L2 cache size, but the fact is, Seaborg L2 cache is an off chip cache in addition to the smaller L1 cache. When the data sizes are larger with the T42 and T85, DataStar P655 and P690 tend to have better coupling than Lemieux. This is due to the presence of the L3 cache in DataStar and its absence in Lemieux. This only appears for larger datasets as data is larger and hence data reuse makes use of lower memory levels. To illustrate, with larger data sizes, data tend to be replaced constantly from L1 cache to L2. Furthermore, by having larger data, more blocks are replaced out of L2. This replacement is more costly on Seaborg and Lemieux than it is the case in DataStar.

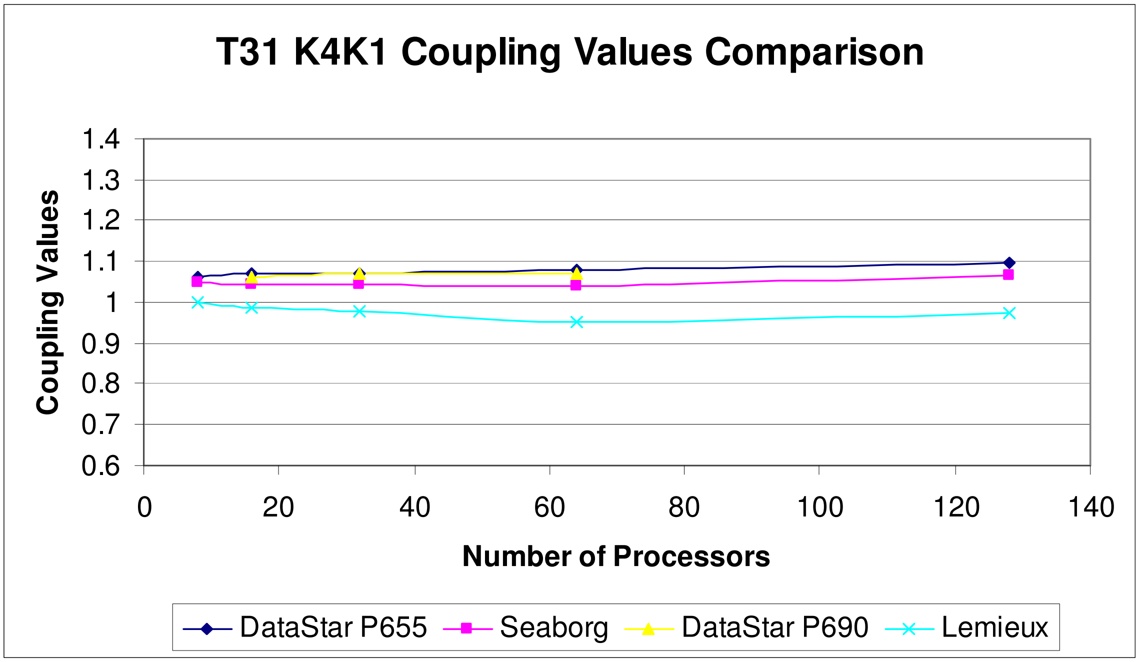

Figure 31: Coupling Values Comparison of the K4-K1 Kernel Pair for the T31 Dataset on the Four Platforms

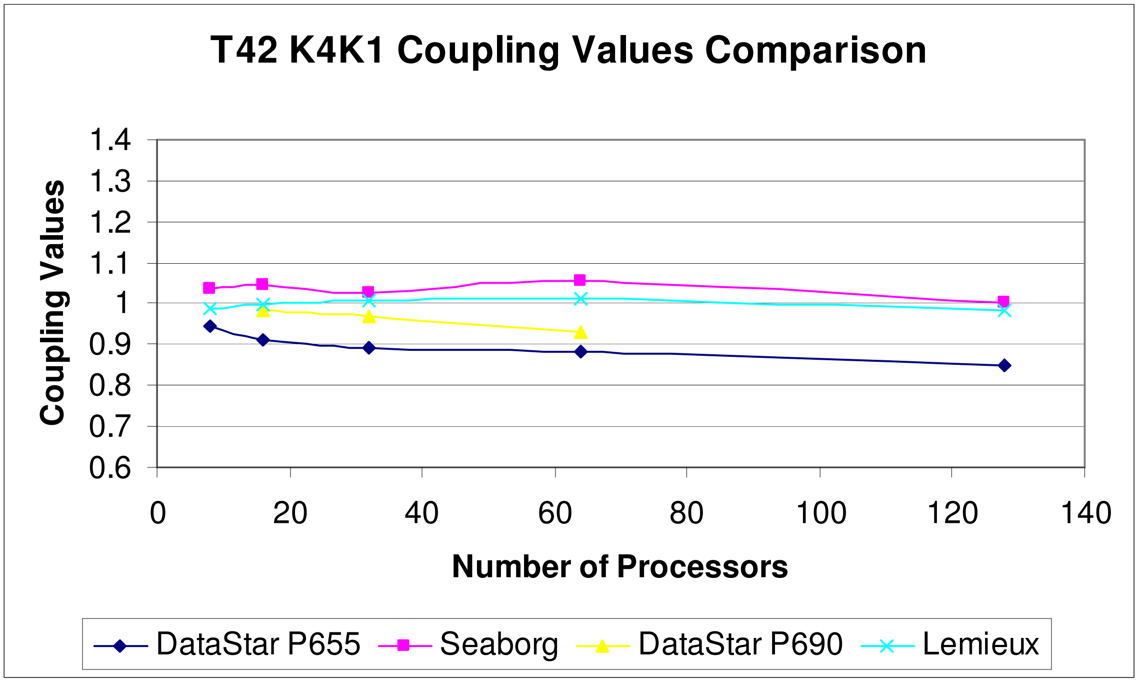

Figure 32: Coupling Values Comparison of the K4-K1 Kernel Pair for the T42 Dataset on the Four Platforms

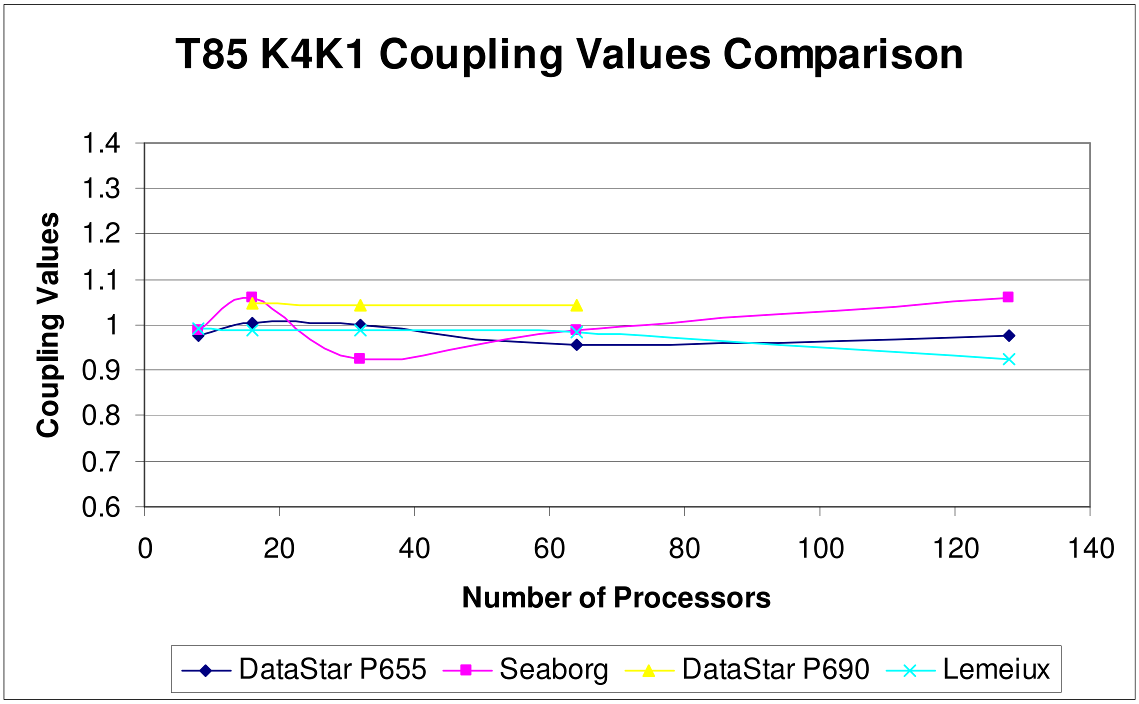

Figure 33: Coupling Values Comparison of the K4-K1 Kernel Pair for the T85 Dataset on the Four Platforms

3.3.1.5 Summary of Kernel Coupling Analysis*

- K1-K2 kernel pair is the only kernel pair with constructive coupling. This was due to the second sub-kernel of K1 where phys_state data structure being the only data structure used. This results in high data reuse as the phys_state structure is the only structure being used by K2. This trend was amplified on Seaborg as it has the 128-way set associative D-Cache that favors the data with high locality and high reuse.

- K2-K3 was the most interesting kernel pair due to the high variation in coupling values from one dataset to another and from one machine to another. Since K2 has the phys_state structure which is the biggest data structure in CAM as indicated by Tables 2 and 3, K3 data was being constantly replaced in the caches by K2 data resulting in high coupling values for K2-K3. This was boosted

* The same analysis done on the kernel pair coupling values can be extended to the three kernels chain coupling values.

because running K3 in isolation was causing K3 data to be residing in caches longer achieving better execution time.

- Lemieux with the largest L1 Cache, and the largest L2 cache (on chip), has a very distinct and stable behavior on all kernel pairs and chains of three kernels. Lemieux is the only platform that didn't experience any DESTRUCTIVE coupling on any dataset and on any number of processors. All the coupling values where either below 1 or approaching 1, which means that all the coupling was either CONSTRUCTIVE or no coupling was taking effect. Thus, the larger cache size was helping the data sharing and data reuse between kernels.

- Since, the inner loop that is iterating over (columns in K2 or longitude in K4) runs sequentially over contiguous memory locations; cache placement policy had some effect on the cache misses. To illustrate, DataStar uses Power4 with 64KB D-Cache 2-way set associate, while Seaborg uses Power3 with 64KB D-Cache 128 way set associative. This different placement policy caused Seaborg to have a better hit rate in some cases over DataStar specially when K2 was involved and the phys_state structure is being used. This is because phys_state having small number of columns per chunk achieves high cache locality.

3.3.2 Performance Prediction

In this work, kernel coupling was used to analyze the interaction between kernels and identify the kernels with the most data sharing and reuse. In this section, K1 refers to D_P_COUPLING kernel, K2 refers to PHYS_PKG kernel, K3 refers to P_D_COUPLING kernel and finally K4 refers to DYN_PKG kernel. Since there were four kernels in CAM, kernel pairs and chains of three kernels had to be tested. In all the runs, kernel pairs or chains of three kernels were executed in a loop of 500 iterations to make sure that the data residing in the caches is the data under test.

Kernel coupling produced a very high error prediction rate for the three datasets on all platforms. The exact prediction values and percentage errors are shown in Tables

15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 and 26. It is clear from the results in the tables that the kernel coupling prediction is very close to summation prediction. This is due to the fact that almost all coupling values are very close to 1 which implies very low coupling between kernels. Furthermore, the results shown in the table indicates that percentage error ranges between 20% and 50% for T42 on Lemieux. This is due to two reasons. The first reason that causes a huge percentage error is the nature of CAM. In CAM, the loop is a time loop that keeps iterating by advancing time. This time advancement can't be captured when running the kernels in isolation otherwise, the model blows up. The second reason is also related to how PHYS_PKG works. The PHYS_PKG does initialization of many variables and data structure during the first time step. When the kernels are run in isolation, the isolation loop is 500 iterations. Thus the average time per kernel is less than the average run per kernel when run in normal execution as in normal execution the maximum number of time steps (iterations) is 148.

Tables 15, 16, 17 show kernel coupling results on Seaborg.

Table 15: T31 Coupling Data on Seaborg

Number of Processors	32	64	128	256	512
Actual Execution Time	42.228717	27.802381	20.226969	19.66108	19.37684
Summation	25.0910378	17.3899696	12.7643395	14.236365	12.31124
Prediction Error	40.58%	37.45%	36.89%	27.59%	36.46%
Prediction using 2 Kernels	25.3571147	17.5858846	13.1115121	14.563855	12.74981
Prediction Error	39.95%	36.75%	35.18%	25.93%	34.20%
Prediction using 3 Kernels	25.4843792	17.7021249	12.994978	14.611104	13.01184
Prediction Error	39.65%	36.33%	35.75%	25.69%	32.85%

Table 16: T42 Coupling Data on Seaborg

Number of Processors	32	64	128	256	512
Actual Execution Time	82.82715	52.357987	37.439146	28.876249	25.77038
Summation	62.2340649	37.2806048	25.8341991	21.754938	24.66825
Prediction Error	24.86%	28.80%	31.00%	24.66%	4.28%
Prediction using 2 Kernels	62.8418427	37.7951448	26.1337012	22.032979	24.81883
Prediction Error	24.13%	27.81%	30.20%	23.70%	3.69%
Prediction using 3 Kernels	62.346939	38.6561293	26.2377473	23.299264	24.72023
Prediction Error	24.73%	26.17%	29.92%	19.31%	4.08%

Table 17: T85 Coupling Data on Seaborg

Number of Processors	32	64	128	256	512
Actual Execution Time	479.904102	276.373765	173.300197	126.59174	101.2833
Summation	393.627459	221.763732	140.9289	99.533172	81.20083
Prediction Error	17.98%	19.76%	18.68%	21.37%	19.83%
Prediction using 2 Kernels	391.163884	224.244118	130.929409	99.946938	82.30398
Prediction Error	18.49%	18.86%	24.45%	21.05%	18.74%
Prediction using 3 Kernels	394.981511	218.461269	134.175221	99.678751	84.83371
Prediction Error	17.70%	20.95%	22.58%	21.26%	16.24%

Tables 18, 19 and 20 show kernel coupling results on DataStar P655.

Table 18: T31 Coupling Data on P655

Number of Processors	16	32	64	128	256
Actual Execution Time	19.40919	11.667294	7.147747	5.849899	5.806424
Summation	9.83713156	5.74112504	3.64931876	3.0537826	3.008282
Prediction Error	49.32%	50.79%	48.94%	47.80%	48.19%
Prediction using 2 Kernels	9.92821329	5.82909874	3.70597131	3.1117138	3.075372
Prediction Error	48.85%	50.04%	48.15%	46.81%	47.04%
Prediction using 3 Kernels	9.96151625	5.82923	3.71471964	3.1162929	3.101381
Prediction Error	48.68%	50.04%	48.03%	46.73%	46.59%

Table 19: T42 Coupling Data on P655

Number of Processors	16	32	64	128	256
Actual Execution Time	40.627136	23.842969	14.878706	10.574727	9.932914
Summation	24.3323089	13.6594541	7.95374366	6.1103434	7.146598
Prediction Error	40.11%	42.71%	46.54%	42.22%	28.05%
Prediction using 2 Kernels	24.4439793	3474.69223	7.92458505	6.2154685	7.058937
Prediction Error	39.83%	14473.24%	46.74%	41.22%	28.93%
Prediction using 3 Kernels	24.6280343	13.543692	7.86980782	6.4770121	7.007105
Prediction Error	39.38%	43.20%	47.11%	38.75%	29.46%

Table 20: T85 Coupling Data on P655

Number of Processors	16	32	64	128	256
Actual Execution Time	258.017587	140.347492	81.874917	49.082112	36.65649
Summation	214.864651	111.749421	62.1261642	41.307122	28.4033
Prediction Error	16.72%	20.38%	24.12%	15.84%	22.51%
Prediction using 2 Kernels	224.42561	115.218067	65.0321343	43.189453	30.49381
Prediction Error	13.02%	17.91%	20.57%	12.01%	16.81%
Prediction using 3 Kernels	223.935036	113.757509	63.7798883	41.069842	28.33422
Prediction Error	13.21%	18.95%	22.10%	16.32%	22.70%

Tables 21, 22 and 23 show kernel coupling results on DataStar P690.

Table 21: T31 Coupling Data on P690

Number of Processors	16	32	64
Actual Execution Time	16.632432	11.832137	10.387081
Summation	8.76674576	5.53778884	5.46890884
Prediction Error	47.29%	53.20%	47.35%
Prediction using 2 Kernels	8.84197341	5.59320359	5.54245519
Prediction Error	46.84%	52.73%	46.64%
Prediction using 3 Kernels	8.87300783	5.60894744	5.56980715
Prediction Error	46.65%	52.60%	46.38%

Table 22: T42 Coupling Data on P690

Number of Processors	16	32	64
Actual Execution Time	34.821296	23.576904	20.179434
Summation	21.0230854	15.0979825	13.3158524
Prediction Error	39.63%	35.96%	34.01%
Prediction using 2 Kernels	21.1682745	15.1173988	13.1740706
Prediction Error	39.21%	35.88%	34.72%
Prediction using 3 Kernels	21.4962481	15.2985529	13.3898455
Prediction Error	38.27%	35.11%	33.65%

Table 23: T85 Coupling Data on P690

Number of Processors	16	32	64
Actual Execution Time	217.645541	143.262683	118.01757
Summation	176.082232	108.160116	85.116305
Prediction Error	19.10%	24.50%	27.88%
Prediction using 2 Kernels	177.698779	109.606703	88.2504969
Prediction Error	18.35%	23.49%	25.22%
Prediction using 3 Kernels	179.230168	110.731592	87.113182
Prediction Error	17.65%	22.71%	26.19%

Tables 24, 25 and 26 show kernel coupling results on Lemieux.

Table 24: T31 Coupling Data on Lemiex

Number of Processors	8	16	32	64	128
Actual Execution Time	63.664651	37.911851	30.926733	22.904117	25.36138
Summation	34.7648687	18.2797708	15.6555909	9.9703296	14.00117
Prediction Error	45.39%	51.78%	49.38%	56.47%	44.79%
Prediction using 2 Kernels	34.3665972	18.0885386	15.4544454	9.7603949	13.85049
Prediction Error	46.02%	52.29%	50.03%	57.39%	45.39%
Prediction using 3 Kernels	34.0623188	18.067575	15.4995863	9.8691539	13.96964
Prediction Error	46.50%	52.34%	49.88%	56.91%	44.92%

Table 25: T42 Coupling Data on Lemieux

Number of Processors	8	16	32	64	128
Actual Execution Time	131.395851	78.541591	53.298564	44.841368	35.54182
Summation	82.5764962	45.6758083	27.0389774	27.800711	17.36372
Prediction Error	37.15%	41.85%	49.27%	38.00%	51.15%
Prediction using 2 Kernels	81.1276917	45.375298	26.9574827	22.601942	17.26327
Prediction Error	38.26%	42.23%	49.42%	49.60%	51.43%
Prediction using 3 Kernels	80.9452431	45.0481333	27.1012465	22.971715	17.35848
Prediction Error	38.40%	42.64%	49.15%	48.77%	51.16%

Table 26: T85 Coupling Data on Lemieux

Number of Processors	8	16	32	64	128
Actual Execution Time	864.965873	483.699936	316.551665	186.22169	141.4743
Summation	721.245311	379.773383	231.479893	111.77632	76.16109
Prediction Error	16.62%	21.49%	26.87%	39.98%	46.17%
Prediction using 2 Kernels	711.0663	377.685943	229.726883	110.02786	73.36528
Prediction Error	17.79%	21.92%	27.43%	40.92%	48.14%
Prediction using 3 Kernels	708.205122	378.000411	228.476971	109.80479	75.08492
Prediction Error	18.12%	21.85%	27.82%	41.04%	46.93%

4. RELATED WORK

Worley and Drake in [6] developed a new implementation for the PHYS_PKG and demonstrated its effect on performance. Their work focused on the modifications done to the PHYS_PKG and how the new CAM design was aiming at decoupling the physics from the dynamics to have CAM compatible with different dynamics. The decision to decouple the physics and dynamics data structures incurred copy overhead and required additional memory, but was justified by the ability to support multiple dynamical cores [6]. Also, they examined how load balancing and the use of OpenMP threads can give similar of not better results than cache blocking.

Mirin and Sawyer in [3] introduced a scalable message passing implementation to the finite volume dynamical core of CAM. Due to the data dependencies resulting from the polar singularity of the latitude-longitude coordinate system, Mirin and Sawyer employed two separate domain decompositions within the dynamical core – one in latitude/level space, and the other in longitude/latitude space. This requires that the data be periodically redistributed between these two decompositions. They used MPI for message passing and OpenMP for multi-threading. They had some executions that scaled to 3000 processors for certain datasets. They also demonstrated the feasibility of nested OpenMP constructs on the IBM, although the net benefit for this particular application is marginal.

5. SUMMARY AND FUTURE WORK

5.1 Summary

This thesis focused on analyzing the Parallel Community Atmosphere Model application. In this analysis several schemes and tools were utilized. We started by analyzing the general behavior of CAM by running it on different machines with different configurations. Through these runs, several characteristics of CAM were identified. Utilizing the Prophesy infrastructure, identifying the runtimes of separate kernels of CAM and their respective scalability was simple. Through this general analysis of CAM behavior the following characteristics and trends were identified:

- There are four major kernels in CAM:
  1 K1: Dynamics to Physics Coupler that is responsible for filling up the data structures used by the physics parameterization package which is mainly the phys_state structure.
  2 K2: Physics Parameterization Package is the kernel responsible for all the physical parameterizations and computations. It is the most dominant kernel in runtime where it dominates over 50% of the overall execution time.
  3 K3: Physics to Dynamics Coupler is the kernel responsible for copying the phys_state structure into the arrays used by the Dynamics package.
  4 K4: The Dynamical Core where all the dynamical computation is done. This is the second dominant kernel in execution time where it dominates approximately 30% of the total execution time of CAM.

- In CAM 3.0, decoupling of the Physical Parameterization from the Dynamical Core was the major advancement. This decoupling of both kernels accomplished two main targets. The first target is allowing CAM to be compatible with more than one Dynamical core. Thus, in CAM 3.0 there are three supported dynamical cores, Eulerian Dynamics, Semi-Lagrangian Dynamics and Finite Volume Dynamics. The second target that was accomplished by this decoupling was allowing researchers to optimize each package separately, the Physics and the Dynamics. This was not achievable before decoupling of data as researchers had to design data structures to be compatible with both and hence their optimization was limited.

- The decoupling of Physics data and Dynamics data had some negative impact on the application behavior. The introduction of the dp_Coupler, Dynamics to Physics Coupler, module boosted CAM's reliance on memory. In both cases, dynamics to physics coupling or physics to dynamics coupling, intensive memory usage is required due to the copying of the data structures from one form to another. As indicated by Tables 2 through 5, the data structures per kernel per processor can be over 200Kbytes for 128 processors. This exceeds the sizes of any D-Cache of any of the supercomputers concerned in this work.

- CAM can support both OpenMP (shared memory) and MPI (message passing) communication. To reach the maximum number of processors possible, CAM is configured to run with certain number of tasks, depending on the dataset, where tasks communicate using MPI. Within each of these tasks OpenMP threads can be utilized to have each thread running on a separate processor reaching the maximum number of processors. The number of tasks is limited by the dataset size. This, in fact, is due to the nature of the data that CAM uses. In atmosphere, computation is independent between grid latitudes. Thus, latitudes are the parallelizable dimension. In this sense, the number of latitudes per dataset is the determining factor of the maximum number of tasks. To illustrate, in T31 the number of latitudes are 16, thus a maximum of 16 tasks is the optimal value. For 32 tasks the model execution starts degrading and for 64 tasks the model blows up.

Also, processor partitioning scheme was used in analyzing the behavior of the MPI only version of CAM. The aim of processor partitioning analysis is to identify the application factors that impact the selection of the best number of processors per node to use for execution of MPI applications. To analyze the performance of CAM, it was executed on DataStar P655, P690, Lemieux and NERSC Seaborg. The total number of processors was kept constant while changing the number of processors per node to see the effect of such configuration. The total runtime, communication time and initialization were collected in order to see the effect on both communication and computation. Initialization was an important factor due to its heavy reliance on I/O. Through this analysis the following was identified about CAM's behavior:

- CAM is very interesting in that the major performance difference occurs with between the scheme utilizing all the processors per node and half of the maximum number of processors per node, with half of the maximum number of processors per node being the better scheme. Further, there is very little difference in the execution time between using one to half of the maximum number of processors per node. When all the processors per node are used, congestion can occur due to data copies of arrays. When half of the maximum number of processors or fewer per node are used the intra-node bandwidth is sufficient

The last scheme used in analyzing the performance of CAM was the kernel coupling scheme. In this work, kernel coupling was used to analyze the interaction between kernels and identify the kernels with the most data sharing and reuse. Since there were four kernels in CAM, kernel pairs and chains of three kernels had to be tested. In all the runs, kernel pairs or chains of three kernels were executed in a loop of 500 iterations to make sure that the data residing in the caches is the data under test. Through kernel coupling the following was identified about CAM's behavior:

- K1-K2 kernel pair is the only kernel pair with constructive coupling. This was due to the second sub-kernel of K1 where phys_state data structure being the only data structure used. This results in high data reuse as the phys_state structure is the only structure being used by K2. This trend was amplified on Seaborg as it has the 128-way set associative D-Cache that favors the data with high locality and high reuse.

- K2-K3 was the most interesting kernel pair due to the high variation in coupling values from one dataset to another and from one machine to another. Since K2 has the phys_state structure which is the biggest data structure in CAM as indicated by Tables 2 and 3, K3 data was being constantly replaced in the caches by K2 data resulting in high coupling values for K2-K3. This was boosted because running K3 in isolation was causing K3 data to be residing in caches longer achieving better execution time.

- Lemieux with the largest L1 Cache, and the largest L2 cache (on chip), has a very distinct and stable behavior on all kernel pairs and chains of three kernels. Lemieux is the only platform that didn't experience any DESTRUCTIVE coupling on any dataset and on any number of processors. All the coupling values where either below 1 or approaching 1, which means that all the coupling was either CONSTRUCTIVE or no coupling was taking effect. Thus, the larger cache size was helping the data sharing and data reuse between kernels.

- Since, the inner loop that is iterating over (columns in K2 or longitude in K4) runs sequentially over contiguous memory locations; cache placement policy had some effect on the cache misses. To illustrate, DataStar uses Power4 with 64KB D-Cache 2-way set associate, while Seaborg uses Power3 with 64KB D-Cache 128 way set associative. This different placement policy caused Seaborg to have a better hit rate in some cases over DataStar specially when K2 was involved and the phys_state structure is being used. This is because phys_state having small number of columns per chunk achieves high cache locality.

5.2 Future Work

In addition to the previous analysis of CAM, there are some new areas to be explored.

- How does the OpenMP only version behave? The only constraint on that version is that it cannot go beyond the node boundary and hence the number of processors will be limited to the maximum number of processors per node.

- CAM can support two other dynamical cores that were not tested in this work. Each of these cores has different data structures and different design. Kernel coupling can be utilized to measure the degree of interaction between the new cores and quantify this interaction. Through comparing the results of coupling values obtained from different kernels, a better design for the dynamical cores can be achieved.

- Enhancing the coupling formula to account for applications where each time step has different runtime than the next. In CAM, the early time steps in the model are used to setup the model and initialize all the data and structures. Thus earlier time steps takes much longer than the average time step. The current formula didn't give an accurate prediction of CAM runtime due to this varying time step.

- Looking into extending the coupling method to utilize the coupling values from multiple chains into one equation to predict performance.

REFERENCES

[1] W. D. Collins, P. J. Rasch, B. A. Boville, J. J. Hack, J. R. McCaa, et al. *Description of the NCAR Community Atmosphere Model (CAM 3.0)*. NCAR TECHNICAL NOTE. June 2004, Retrieved June 12, 2006 from http://www.ccsm.ucar.edu/models/atm-cam/docs/description/

[2] J. Geisler, V. Taylor, X. Wu, and R. Stevens, Using Kernel Coupling to Improve the Performance of Multithreaded Applications, In *Proc. of the 16th International Conference on Parallel and Distributed Computing Systems (PDCS-2003)*, Reno, Nevada, August 13-15, 2003

[3] A. A. Mirin and W. B. Sawyer A scalable implementation of a finite-volume dynamical core in the Community Atmosphere Model. *International Journal of High Performance Computing Applications,* 19(3), August 2005, pp. 203-212

[4] Network Common Data Form, (n.d), Retrieved June 12, 2006 from http://www.unidata.ucar.edu/software/netcdf/

[5] V. Taylor, X. Wu, J. Geisler, and R. Stevens, Using Kernel Couplings to Predict Parallel Application Performance, In *Proc. of the 11th IEEE International Symposium on High-Performance Distributed Computing (HPDC 2002)*, Edinburgh, Scotland, July 24-26, 2002

[6] P. H. Worley and J. B. Drake Performance Portability in the Physical Parameterizations of the Community Atmosphere Model, *International Journal for High Performance Computer Applications*, 19(3), August 2005, pp. 1-15.

[7] X. Wu, V. Taylor, C. Lively, and S. Sharkawi Processor Partitioning: An Experimental Performance Analysis for MPI Applications, submitted to *International Conference for High Performance Computing, Networking, Storage and Analysis (SC06)*, Tampa, Florida, November 2006.

[8] X. Wu, V. Taylor, J. Geisler, and R. Stevens, Isocoupling: Reusing Coupling Values to Predict Parallel Application Performance, In *Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS2004)*, Santa Fe, New Mexico, April 26-30, 2004

VITA

NAME: Sameh Sherif Shawky Sharkawi

ADDRESS: 301 Harvey R. Bright Building, College Station, TX 77843-3112

EMAIL ADDRESS: sss1858@cs.tamu.edu

EDUCATION: B.S., Computer Science, The American University in Cairo, 2002

M.S., Computer Science, Texas A&M University, 2006