Automatic Performance Prediction for Load-Balancing Coupled Models

Daihee Kim*, J. Walter Larson†, and Kenneth Chiu*
*State University of New York at Binghamton
{dkim17, kchiu}@cs.binghamton.edu
†The Australian National University
jay.larson@anu.edu.au

***Abstract*—Computationally-demanding, parallel coupled models are crucial to understanding many important multiphysics/multiscale phenomena. Load-balancing such simulations on large clusters is often done through off-line, static means that often require significant manual input. Dynamic, runtime load-balancing has been shown in our previous work to be effective, but we still used a manually generated performance predictor to guide the load-balancing decisions. In this paper, we show how timing and interaction information obtained by instrumenting the middleware can be used to automatically generate a performance predictor that relates the overall execution time to the execution time of each individual submodel. The performance predictor is evaluated through the new coupled model benchmark employing five constituent submodels that simulates the CCSM coupled climate model.**

***Keywords*-MPI, Dynamic Load Balance, Model Coupling, Multiphysics Modeling, Multiscale Modeling**

I. Introduction

The behavior of complex systems often arise from *multiphysics* and *multiscale* interactions between different subsystems. For example, climate depends on interactions between subsystems such as the ocean and the atmosphere. Typically, these system are simulated by developing a model for each subsystem and coupling the models together, resulting in what are commonly known as *coupled* models. To distinguish from the model itself (which is often expressed mathematically and/or conceptually), we call the computational entity responsible for each submodel a *constituent*.

These coupled models can be computationally intensive, and thus are often parallelized and run on large clusters. Scaling a coupled model to such a cluster, however, requires appropriate allocation of resources—both in processing element (PE) space and in time—to constituents. Many coupled models, such as those for climate modeling, are fundamentally synchronous. Each model must periodically synchronize (and exchange data) with others at fixed intervals. Thus, a model constituent that runs faster than the others may block while waiting for other constituents to catch up. For optimal performance, resources should thus be load-balanced, i.e., allocated such that there is minimal blocking of one constituent by another. The load-balancing problem is difficult, however, because the run-time performance interactions between constituents are complex.

Coupled model developers often employ static load-balancing that is determined in a manual, ad hoc manner. This process is laborious, tedious, and time-consuming; and is a platform-dependent tuning problem. Moreover, there is no guarantee that a configuration determined this way will perform equally well as the system executes since the computational load in a constituent may change during execution.

Another approach is to use a dynamic load-balancing technique with limited a priori knowledge about the performance characteristics of the coupled model. This approach requires that coupled models have the ability to dynamically reallocate computational resources at runtime, a characteristic known as *malleability* [1], and we call a coupled model that allows dynamic interconstituent load balance a *malleable coupled model* (MCM). Re-allocations are guided by measured runtime performance to adjust the resource allocation dynamically, seeking to optimize performance. Efficiently using this approach, however, requires that we be able to predict how a change to a given constituent's execution time will change the overall execution time of a coupled model. This is challenging, because there are complex communication and computation interactions.

We have previously shown a dynamic load-balancing approach using a manually-generated heuristic model to do this prediction [2], [3]. The model was statically and manually created by inspecting and understanding the coupled model interaction patterns and constituents' computations, and is thus impractical for a more complex, realistic coupled model. In this paper, we show how a performance model for coupled models can be generated *automatically* and *dynamically* by instrumenting each constituent. The generation procedure uses measurement data continuously collected from previous iterations during the execution. Predictions from the automatically generated prediction model are then used to guide load-balancing decisions. Furthermore, we optimize the overhead associated with the interaction between coupled model and our load-balancing manager. We also present a new coupled model benchmark composed of five constituents that mimics the Community Climate System Model (CCSM3) [4].

II. Related Work

The strategy of employing resource malleability to load-balancing of parallel iterative applications has been investigated by others. Ko et al. [5] introduced a coupled multiphysics simulation system that is able to optimize PE cohorts through runtime PE reallocation. Their load-balancing algorithm, however, is applicable only to a coupled model with two constituents and with the assumption that computation time is reduced ideally by parallelization.

The Stop Restart Software (SRS) library [6] allows a parallel application to reconfigure PE cohorts by stopping and restarting its execution. El Maghraoui et al. [7], [8] have developed a Process Checkpointing and Migration (PCM) library, which, together with their Internet Operating System (IOS) constitute a framework to support malleable, iterative MPI applications. The Resizing and Scheduling of Homogeneous Applications in a Parallel Environment (ReSHAPE) [9] framework changes PE allocations of malleable parallel applications during job scheduling for system resource utilization. Utera et al. [10] also dealt with malleability for efficient job scheduling. None of these approaches, however, are immediately applicable to malleable model coupling because they concentrate on applying malleability to monolithic parallel applications; that is, the aforementioned approaches fail to account for the sensitivity of a coupled system to its interconstituent data dependencies.

The CSCAPES project [11] has as one of its foci dynamic load-balancing for parallel applications. Hypergraph-based repartitioning with Zoltan [12], one of CSCAPES contributions, performs dynamic load-balancing through data/computation migration within a model's PE cohort. Charm++ and Adaptive MPI (AMPI) [13], [14] allow to develop and execute parallel application based on migratable data/computation objects interacting via asynchronous message invocation for dynamic load-balancing. Uintah [15], [16] is a framework supporting development and execution of components consisting of tasks with load-balancing application such as Zoltan. However, model coupling is not allowed unless the individual submodels are initially designed and developed based on their approaches. Moreover, these approaches require significant effort to distinguish and compose fine-grained computation entities over all subsystems to build a large multiphysics model.

Performance of the Community Earth System Model (CESM) (the new release of CCSM) and its resource allocation has been investigated in [17]. However, the paper describes performance optimization for CESM to obtain better throughput with given PEs by analysis of parallel computational properties of constituents and trial and error, without an automatic optimization scheme.

III. Malleable Model Coupling and MMCT

The state of a coupled model is the union of its constituents' states, which are computed by their respective subsystem models. Each constituent solves its governing equations using (providing) boundary and forcing data from (to) its peers during *coupling events*. If all of a system's interconstituent couplings are scheduled, have interval times that are mutually commensurate, and fit within a repeatable time window of size ΔT, the system has a *coupling cycle* with *period* ΔT. The quantity ΔT is the minimum time interval during which all the system's interconstituent data dependencies have been encountered, and is the coupled system's overall "timestep."

Load-balancing of a coupled model aims to minimize the total amount of idle time across its subsystem constituents' PE pools. The object is to do this and maximize overall system throughput by minimizing the wall-clock *global iteration time* τ_G required to evolve the coupled system through a period ΔT. Each constituent expends a wall-clock *constituent iteration time* τ_i to integrate its respective equations of evolution for the coupling period, including intra- and inter-constituent communications costs. The iteration time τ_i has an additive decomposition in terms of the *constituent computation time* τ_i^{comp} and *constituent coupling time* τ_i^{coup}; that is, $\tau_i = \tau_i^{comp} + \tau_i^{coup}$ [18]. The quantity τ_i^{coup} signifies the wall-clock time that ith constituent spends and awaits to complete sending (receiving) data to (from) other constituents. The quantity τ_i^{comp} is the wall-clock time required by the ith constituent to compute its internal state, *including* any intra-constituent communication. The load-balancing problem becomes one of analyzing values of $(\tau_i^{comp}, \tau_i^{coup}), i = 1, \ldots, N$ for a variety of PE cohort size configurations to optimize $\tau_G = max\{\tau_i, \ldots, \tau_N\}, i = 1, \ldots, N$. This requires re-sizing of PE cohorts—malleability—to shift load allocation.

We extended the Model Coupling Toolkit (MCT; [19], [20]), resulting in the Malleable Model Coupling Toolkit (MMCT) [18] supporting runtime load-balancing in parallel coupled models. MMCT provides runtime support for interconstituent PE reallocation and global PE cohort (i.e., MPI_COMM_WORLD) resizing. This is accomplished by two extensions to MCT: a dynamic process and communicator management system (PCMS) and a centralized load balance manager (LBM). The LBM communicates with each constituent's head node and it gathers and analyzes throughput information—values of τ_G and $(\tau_i^{comp}, \tau_i^{coup}), i = 1, \ldots, N$—to make PE cohort reallocation decisions; its analyses are guided by a load-balancing algorithm. The PCMS implements the LBM's decisions by employing dynamic process creation/termination. MMCT performs these operations over a predefined load balance interval (LBI) corresponding to an integral multiple of ΔT. At the end of each LBI, each constituent will increase, decrease, or preserve the number of PEs in its cohort. Further details on a runtime architecture and the software implementation of MMCT can be found in [18], [2].

The LBM's load-balancing algorithm aims to reduce τ_G by taking multiple PEs from one or more *donor* constituents and giving them to one or more *recipient* constituents. Its decisions are guided by current measurements of and predictions for future values of τ_i^{comp} and τ_G. We have developed a simple heuristic τ_G prediction model that should be defined statically and manually and use predicted τ_i^{comp} [2]. This system has since been enhanced to allow time-variant performance curve of τ_i^{comp} and τ_G [3].

IV. Performance Prediction Model

The quantity we wish to maximize—coupled model throughput—is inversely proportional to the global iteration time τ_G. The two factors that most strongly affect τ_G are constituent scalability in the absence of interconstituent communication and interconstituent computation serialization imputed by interconstituent communications traffic. We combine

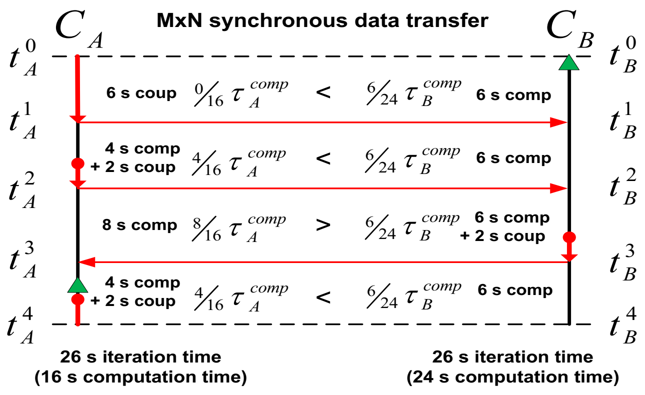

MxN synchronous data transfer
6 s coup
4 s comp + 2 s coup
8 s comp
6 s comp + 2 s coup
6 s comp
26 s iteration time (16 s computation time)
26 s iteration time (24 s computation time)

Fig. 1. Effect of coupling-imputed serialization on τ_G for a two-constituent coupled system. Left (right) vertical bar represents timeline for constituents A and B over a coupling cycle spanning $[t_A^0, t_A^4]$ ($[t_B^0, t_B^4]$). Each constituent's respective time loop is begun at points indicated by green triangles. Horizontal red arrows indicate $M \times N$ synchronous data transfer; arrow direction indicates direction of data flow. Vertical red arrows indicate constituent execution time consumed by blocking $M \times N$ send operation invoked at points indicated by red circles. For this example, $\tau_G = \tau_A = \tau_B = 26s$.

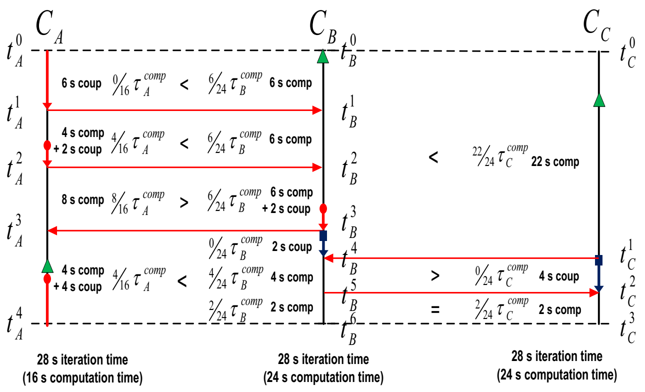

6 s coup
4 s comp +2 s coup
8 s comp
6 s comp +2 s coup
2 s coup
4 s comp +4 s coup
4 s comp
2 s comp
22 s comp
4 s coup
2 s comp
28 s iteration time (16 s computation time)
28 s iteration time (24 s computation time)
28 s iteration time (24 s computation time)

Fig. 2. The other example for computation serialization between three constituents A, B and C. Vertical blue arrows indicate constituent execution time consumed by blocking $M \times N$ receive operation invoked at points indicated by blue rectangles. For this example, $\tau_G = 28s$.

models for each of these factors, constrained by in situ timing measurements, to arrive at our performance prediction model.

Our model for parallel scalability of a constituent $\mathcal{C}_i$ in the absence of coupling derives from classical assumptions for scaling behavior of parallel applications. We first assume $\tau_i^{comp} = f_i(p_i)$; f_i is a function of p_i, the number of PEs on which the $\mathcal{C}_i$ executes. We assume f_i decreases monotonically with p_i for $p_i < p_i^*$; scaling saturates at $p_i = p_i^*$ and f_i has its minimum at $p_i = p_i^*$. For $p_i > p_i^*$, f_i monotonically increases, most commonly due to a drop in the ratio of computational to communication intensities.

Coupling-imputed serialization complicates performance modeling substantially. In the analysis that follows, we assume a multiprocessor whose interconnect has low latency $\lambda \to 0$ and high bandwidth $\beta \to \infty$; this assumption holds as well as the conditions $\lambda \ll \tau_G$ and $MS/\beta \ll \tau_G$, where M is the average number of serialized messages in a coupling cycle and S is their average size. Coupled models are typically designed to minimize communicative coupling overhead as a percentage of total execution time; for CESM this fraction is less than 5–10%. Thus, the low latency/high bandwidth régime we assume is appropriate. In sum, we are treating the load imbalances as the leading order effect on τ_G and per-PE $M \times N$ transfer costs as a higher-order effect.

We also assume that within each constituent $\mathcal{C}_i$, its computations occurring within intervals between coupling events within the coupling cycle scale equally well; scaling disparities for $\mathcal{C}_i$'s processing activities within a coupling cycle are treated as a higher-order effect. These assumptions simplify the performance modeling problem to one involving the values of τ_i^{comp} and their measurements.

Refinements to include $M \times N$ transfer effects require at a minimum measurement of the coupling overhead τ_i^{coup}, but may also require detailed knowledge of how $M \times N$ message patterns change w.r.t. p_i. Including intraconstituent scaling disparities for calculations occurring between coupling events would require additional timing measurements for each computational interval. These extensions, while feasible, are complex, and deferred as areas for future investigation.

To elucidate our methodology, we present two examples of coupled models that exhibit coupling-imputed serialization, and how we quantify its effect on τ_G. We will then derive from these motivating examples a general model for τ_G.

Figure 1 shows a simple coupled model with coupled constituents $\mathcal{C}_A$ and $\mathcal{C}_B$ in parallel composition. The vertical dimension in Figure 1 represents time and the left (right) vertical line corresponds to the execution timeline—t_A (t_B)—for constituent $\mathcal{C}_A$ ($\mathcal{C}_B$) over one coupling cycle. Both $\mathcal{C}_A$ and $\mathcal{C}_B$ employ $M \times N$ synchronous data transfers that block overall to keep execution synchronous with respect to their coupling cycle. Suppose the times t_A^0 (t_A^4) and t_B^0 (t_B^4) indicate the beginning (end) of $\mathcal{C}_A$'s and $\mathcal{C}_B$'s time measurements at the same time within a coupling cycle. Interconstituent data transfer events seen by $\mathcal{C}_A$ ($\mathcal{C}_B$) occur at times t_A^1 (t_B^1), t_A^2 (t_B^2), and t_A^3 (t_B^3), partitioning the coupling cycle into four paired time intervals—$\{[t_A^0, t_A^1], [t_B^0, t_B^1]\}$, $\{[t_A^0, t_A^2], [t_B^0, t_B^2]\}$, $\{[t_A^0, t_A^3], [t_B^0, t_B^3]\}$, and $\{[t_A^0, t_A^4], [t_B^0, t_B^4]\}$—for which imputed serialization effects must be identified and analyzed. We urge the reader to examine the respective interval in Figure 1 to each stage of the analysis presented below.

The first interval $\{[t_A^0, t_A^1], [t_B^0, t_B^1]\}$ concludes when $\mathcal{C}_A$'s send operation is completed. Here the bottleneck is $\mathcal{C}_B$'s computation time of 6 seconds, forcing $\mathcal{C}_A$ to wait until $\mathcal{C}_B$ executes its corresponding receive operation. During the second interval $\{[t_A^0, t_A^2], [t_B^0, t_B^2]\}$, if we replace $\mathcal{C}_A$'s computation time during the first interval with the bottleneck time caused by $\mathcal{C}_B$ (6 seconds), $\mathcal{C}_B$ is again the bottleneck in this interval—6 seconds of computation time versus 4 seconds for $\mathcal{C}_A$—resulting in 2 seconds' coupling time. During the interval $\{[t_A^0, t_A^3], [t_B^0, t_B^3]\}$, $\mathcal{C}_A$ is the bottleneck because its compute time of 8 seconds exceeds the $\mathcal{C}_B$'s 6 seconds, creating a coupling delay in B of 2 seconds. The interval $\{[t_A^0, t_A^4], [t_B^0, t_B^4]\}$ has $\mathcal{C}_B$ as the bottleneck with 6 seconds of computation time.

Combining our previous assumptions with the above imputed serialization analysis, we model τ_G as a function of $(\tau_A^{comp}, \tau_B^{comp})$. This model will, in turn, when combined with individual performance models for $\mathcal{C}_A$ and $\mathcal{C}_B$, allow us to predict τ_G as a function of (p_A, p_B). Invoking the assumption of equivalent intraconstituent scaling throughout the coupling cycle and using the given values $\tau_A^{comp} = 16$ s and $\tau_B^{comp} = 24$ s, we can scale the individual interval computation times for $\mathcal{C}_A$ and $\mathcal{C}_B$, respectively, to estimate the total computational work performed in each interval. The interval-by-interval computation serialization analysis yields an estimator for the global iteration time – $\xi(\tau_G)$:

$$\xi_A(t_A^1) = \xi_B(t_B^1) = \max\left\{\frac{0}{16}\tau_A^{comp}, \frac{6}{24}\tau_B^{comp}\right\} \tag{1}$$

$$\xi_A(t_A^2) = \xi_B(t_B^2) = \max\left\{\xi_A(t_A^1) + \frac{4}{16}\tau_A^{comp}, \xi_B(t_B^1) + \frac{6}{24}\tau_B^{comp}\right\} \tag{2}$$

$$\xi_A(t_A^3) = \xi_B(t_B^3) = \max\left\{\xi_A(t_A^2) + \frac{8}{16}\tau_A^{comp}, \xi_B(t_B^2) + \frac{6}{24}\tau_B^{comp}\right\} \tag{3}$$

$$\begin{aligned}\xi(\tau_G) &= \xi_A(t_A^4) = \xi_B(t_B^4)\\ &= \max\left\{\xi_A(t_A^3) + \frac{4}{16}\tau_A^{comp}, \xi_B(t_B^3) + \frac{6}{24}\tau_B^{comp}\right\}\\ &= \frac{6}{24}\tau_B^{comp} + \frac{6}{24}\tau_B^{comp} + \frac{8}{16}\tau_A^{comp} + \frac{6}{24}\tau_B^{comp}\\ &= \frac{1}{2}\tau_A^{comp} + \frac{3}{4}\tau_B^{comp}\end{aligned} \tag{4}$$

Note that $\xi(\tau_G)$ can be also the estimator for time period of $\{[t_A^0, t_A^4], [t_B^0, t_B^4]\}$. Moreover, $\xi_A(t_A)$ and $\xi_B(t_B)$ are estimators for time periods of $[t_A^0, t_A]$ and $[t_B^0, t_B]$ where $t_A^0 \leq t_A \leq t_A^4$ and $t_B^0 \leq t_B \leq t_B^4$ as seen from the perspectives of A and B, respectively. The terms on the RHS of Eqns. (1–4) correspond to and are listed in the same order as the time periods from Figure 1. Substituting the values of τ_A^{comp} and τ_B^{comp} into Eqns. (1–4) yields $\xi(\tau_G) = 26$ s.

Figure 2 shows a system in which $\mathcal{C}_A$ and $\mathcal{C}_B$ from the first example are coupled to a third constituent $\mathcal{C}_c$. Below, we identify and quantify this system's coupling-imputed serializations and then estimate τ_G. During the interval $\{[t_B^0, t_B^4], [t_C^0, t_C^1]\}$, $\mathcal{C}_C$ is bottleneck since $\mathcal{C}_B$ must wait until $\mathcal{C}_C$ performs its send operation, introducing 2 seconds of coupling time to $\mathcal{C}_B$. During the interval $\{[t_B^0, t_B^5], [t_C^0, t_C^2]\}$, $\mathcal{C}_B$ is the bottleneck constituent with 4 seconds of computation time. Finally, $\mathcal{C}_B$ (or $\mathcal{C}_C$) is the bottleneck, with 2 seconds of computation time during the interval $\{[t_A^0, t_A^4], [t_B^0, t_B^6], [t_C^0, t_C^3]\}$. Given the values $\tau_A^{comp} = 16$ s, $\tau_B^{comp} = 24$ s, and $\tau_C^{comp} = 24$ s the estimator for τ_G is:

$$\xi_B(t_B^4) = \xi_C(t_C^1) = \max\left\{\xi_B(t_B^3) + \frac{0}{24}\tau_B^{comp}, \frac{22}{24}\tau_C^{comp}\right\} \tag{5}$$

$$\xi_B(t_B^5) = \xi_C(t_C^2) = \max\left\{\xi_B(t_B^4) + \frac{4}{24}\tau_B^{comp}, \xi_C(t_C^1) + \frac{0}{24}\tau_C^{comp}\right\} \tag{6}$$

$$\begin{aligned}\xi(\tau_G) &= \xi_A(t_A^4) = \xi_B(t_B^5) = \xi_C(t_C^3)\\ &= \max\left\{\xi_A(t_A^3) + \frac{4}{16}\tau_A^{comp}, \xi_B(t_B^4) + \frac{2}{24}\tau_B^{comp}, \right.\\ &\qquad \left. \xi_C(t_C^2) + \frac{2}{24}\tau_C^{comp}\right\}\\ &= \frac{22}{24}\tau_C^{comp} + \frac{4}{24}\tau_B^{comp} + \frac{2}{24}\tau_C^{comp}\\ &= \tau_C^{comp} + \frac{1}{6}\tau_B^{comp}.\end{aligned} \tag{7}$$

Substituting the values of τ_A^{comp}, τ_B^{comp}, and τ_C^{comp} into Eqns. (5–7) yields $\xi(\tau_G) = 28$ s.

The estimation scheme embodied in Eqns. (1–7) can readily be generalized for a model with N constituents $\{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ and a coupling cycle in which each constituent $\mathcal{C}_i$ experiences S_i synchronization points during the coupling cycle. The imputed synchronization points are numbered from 0 where time measurement is begun to S_i where time measurement is ended ($0 \leq t_i \leq S_i$). Following the same arguments regarding fractional serialization, we arrive at the estimator for τ_G, $\xi(\tau_G)$:

$$\xi_i(t_i) = \begin{cases} 0 \quad \text{if } t_i = 0 \\ \max\left\{\xi_i(t_i - 1) + \theta_{(i,t_i)}\tau_i^{comp}, \right. \\ \qquad \left. \xi_j(t_j - 1) + \theta_{(j,t_j)}\tau_j^{comp}\right\} \quad \text{if } t_i < S_i \end{cases} \tag{8}$$

$$\begin{aligned}\xi(\tau_G) &= \xi_i(S_i)\\ &= \max\left\{\xi_1(S_1 - 1) + \theta_{(1,S_1)}\tau_1^{comp}, \right.\\ &\qquad \ldots,\\ &\qquad \left. \xi_N(S_N - 1) + \theta_{(N,S_N)}\tau_N^{comp}\right\}\end{aligned} \tag{9}$$

In Eqns. (8–9), $\theta_{(i,t_i)} = \tau_i^{comp}(t_i)/\tau_i^{comp}$, where $\tau_i^{comp}(t_i)$ is the amount of wall-clock computation time used by $\mathcal{C}_i$ in the interval $[t_i - 1, t_i]$. The estimation formulae (Eqns.(8–9)) allow prediction of a coupled model's performance automatically, dynamically, and without a priori knowledge.

We implemented the estimation formulae (8–9) in MMCT's LBM as follows: For the kth LBI we define the PE layout vector $\vec{P^k} = (p_1^k, \ldots, p_N^k)$, as the number of PEs in each constituent's cohort. Let τ_G^k and $\{\tau_1^{(comp,k)}, \ldots, \tau_N^{(comp,k)}\}$ be the global and constituents' respective iteration times for the kth LBI. The LBM collects all timings including the global iteration time and constituent iteration times and $\tau_i^{(comp,k)}(t_i)$ of all constituents at each LBI. Under our assumptions and using all timing measurements at kth LBI, we can compute the fractions $\theta_{(i,t_i)}^k$ for all constituents. Thus, a performance model for τ_G^k can be generated using the estimator for τ_G Eqn. (9):

$$\tau_G^k = \alpha^k \xi_i(S_i, \theta^k, \tau_i^{(comp,k)}) \tag{10}$$

The factor α^k in Eqn. (10) is included as a tunable parameter to account for factors such as network latency, bandwidth, and timing resolutions. We determine α^k by comparing modeled and measured values of τ_G^k.

Eqn. (10) can be rendered prognostic to compute τ_G^{k+1} if we assume that the values of $\theta_{(i,t_i)}^k$ and α^k are time-invariant (or change slowly with time) and we are capable of accurately modeling $\tau_i^{(comp,k+1)}$ as a function of p_i^{k+1}. Under these assumptions, the automatically generated prediction model for τ_G^{k+1} is:

$$\tau_G^{k+1} = \alpha^k \xi_i(S_i, \theta^k, \tau_i^{(comp,k+1)}) \tag{11}$$

V. Predictive Load-Balancing Algorithm

Here, we concisely describe the load-balancing algorithm presented in previous work and modifications to employ the automatic prediction model and optimize the overhead of interaction between the LBM and coupled model.

```
{Interval $[t_i, t_i^{'}]$ is an amount of time between two time points $t_i$ and $t_i^{'}$, that is the sum of computation and coupling times representing [comp, $t_i, t_i^{'}$] and [coup, $t_i, t_i^{'}$] respectively.}
{$B_i$, $F_i$, $L_i$, and $E_i$ indicate time points where the beginning of time loop, the first and last data transfer, and the end of time loop for $\mathcal{C}_i$ }
repeat
  isrepeat = false;
  for $i = 1$ to $N$ do
    $\{j, t_j\}$ = find $\mathcal{C}_j$ and point $t_j$ communicating with $\mathcal{C}_i$ at $F_i$;
    if $[B_i, F_i] > [B_j, t_j]$ then
      sub = $[B_i, F_i] - [B_j, t_j]$;
      res = [comp, $B_i, F_i$] $-$ sub;
      if res $> 0$ then res = 0; end if
      sub += res;
      [comp, $B_i, F_i$] −= sub; [comp, $L_i, E_i$] += sub;
      [coup, $B_i, F_i$] −= abs(res); [coup, $L_i, E_i$] += abs(res);
      isrepeat = true;
    else if $[B_i, F_i] < [B_j, t_j]$ then
      sub = $[B_j, t_j] - [B_i, F_i]$;
      res = [comp, $L_i, E_i$] $-$ sub;
      if res $> 0$ then res = 0; end if
      sub += res;
      [comp, $B_i, F_i$] += sub; [comp, $L_i, E_i$] −= sub;
      [coup, $B_i, F_i$] += abs(res); [coup, $L_i, E_i$] −= abs(res);
      if [coup, $L_i, E_i$] $< 0$ then
        [coup, $L_i, E_i$] = 0;
      end if
      isrepeat = true;
    end if
  end for
until isrepeat is true
```

Fig. 3. A method to convert the loop based timing result to the coupling cycle based timing result by adjusting computation time and coupling time in two time intervals of all constituents.

The load-balancing algorithm employed by the LBM begins with an initial PE allocation $\vec{P}^0$ to the system's N constituents.

The algorithm attempts to balance load by PE reallocation $\vec{P}^k \rightarrow \vec{P}^{k+1}$ across constituents at every LBI to determine better PE cohort allocations. Reallocation is guided by prediction of τ_G that proceeds by first predicting $\tau_i^{(comp,k+1)}$ and then using $\tau_i^{(comp,k+1)}$ with the prediction model described in Section IV to predict τ_G^{k+1}. Load balance decisions based on predicted τ_G values are made in an *optimization* phase.

A. Predicting τ_i^{comp} and τ_G

Values of τ_i^{comp} are measured in each LBI and stored with their corresponding number of PEs as points in a plane, with p_i (τ_i^{comp}) as the ordinate (abscissa). We employed modified piecewise linear interpolation to predict τ_i^{comp}. Analysis for each constituent is performed in its respective (p_i, τ_i^{comp}) space. The modified linear interpolation algorithm is performed using two regions where the domain is divided at $p_i = \frac{1}{2}\max\{p_i^0, \ldots, p_i^k\}$. Details of the algorithm are described in [2].

We use the automatic prediction model Eqn. (11) in Section IV to predict τ_G using estimated values of $\tau_i^{(comp,k+1)}$. Consider that each constituent begins and ends its own time loop at points unsynchronized with others and starts and stops timing measurement at the beginning and end of the time loop. To convert this time loop based timing data to the coupling cycle based timing result shown in Figure 1 and 2, the LBM collects timing data and has to adjust amounts of computation time and coupling time of each constituent within two time intervals between the time loop beginning point and the first data transfer point and between the last data transfer point and time loop end point, representing $[B_i, F_i]$ and $[L_i, E_i]$ respectively, through a method described in Figure 3. Then the LBM develops the prediction model Eqn. (11) using the current coupling cycle based timing values automatically.

B. Optimization Algorithm

A PE cohort configuration $\vec{P}^k \in \mathbb{N}^N$ is reallocated to $\vec{P}^{k+1}$ corresponding to the vector difference $\Delta\vec{P}^{k+1} = \vec{P}^{k+1} - \vec{P}^k$ that defines the direction of the reallocation in PE space. Donor (Recipient) constituents in a reallocation $\vec{P}^k \rightarrow \vec{P}^{k+1}$ are identified by negative (positive) values of $p_i^{k+1} - p_i^k$; unchanged allocations correspond to $p_i^{k+1} = p_i^k$.

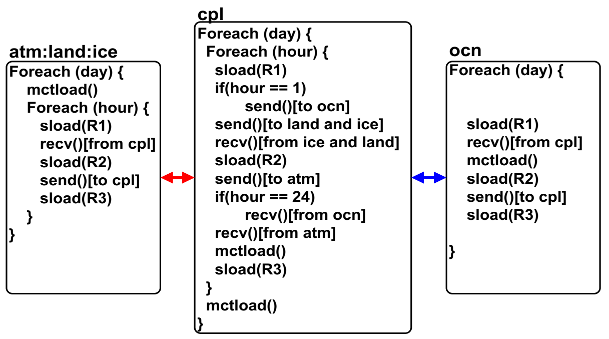
The algorithm basically tries to select a direction corresponding to a previously untried PE allocation, expecting $\tau_G^{k+1} \leq \tau_G^k$ using the τ_G prediction model and predicted τ_i^{comp}. It considers all possible reallocations (directions) and uses the *prediction counter* $\widehat{\Delta P}$ which limits the number of possible PE reallocations (directions); that is, $|\Delta\vec{P}^{k+1}| \leq \widehat{\Delta P}$. If the previous direction fails, the previous reallocation is undone to recover the previous PE allocation. The prediction counter is increased (decreased) if previous direction successes (fails). The algorithm is converged if there are no more possible directions to try. However, it operates again when load changes are detected even though it has converged previously.

Details of the optimization algorithm employing the manually determined prediction model that is replaceable with the automatically generated prediction model are presented in [2], [3]. Note that the manual model includes multiple τ_G estimators; however, the automatic model has only one τ_G estimator.

We also optimize and reduce the LBI, which is the number of coupling cycles required for a reallocation, from four cycles in our previous work to two coupling cycles. The first cycle must be used to overlap constituents' computation for the beginning of a timing cycle. The second cycle is used for timing. The reallocation can then occur at the beginning of the next cycle that can also be the subsequent LBI's first cycle. All constituents wait for reallocation instructions from the LBM at the beginning of every LBI's first cycle for reallocation until load-balancing is converged. Note that the fewer coupling cycles LBM optimizer requires to complete load-balancing, the more rapidly coupled model throughput is improved.

VI. Performance Studies

To validate and evaluate the LBM optimizer with the dynamic and automatic performance prediction model, we developed and used a coupled model benchmark with a varying amount of simulated computation. The benchmark emulates CCSM3, a well known parallel coupled model and has atmosphere (atm), ocean (ocn), land (land), ice (ice), and coupler (cpl) constituents (Figure 4). The atm, ocn, land, and ice exchange interfacial flux and state data through $M \times N$ data transfers with cpl. The cpl constituent has grid information for the other constituents and performs intergrid interpolation of



```
atm:land:ice
Foreach (day) {
    mctload()
    Foreach (hour) {
      sload(R1)
      recv()[from cpl]
      sload(R2)
      send()[to cpl]
      sload(R3)
    }
}
```

```
cpl
Foreach (day) {
  Foreach (hour) {
    sload(R1)
    if(hour == 1)
          send()[to ocn]
    send()[to land and ice]
    recv()[from ice and land]
    sload(R2)
    send()[to atm]
    if(hour == 24)
          recv()[from ocn]
    recv()[from atm]
    mctload()
    sload(R3)
  }
  mctload()
}
```

```
ocn
Foreach (day) {


    sload(R1)
    recv()[from cpl]
    mctload()
    sload(R2)
    send()[to cpl]
    sload(R3)
}
```

Fig. 4. Climate benchmark application using sload(R). Intercommunication occurs 24 times between atm, land, ice and cpl and once between ocn and cpl within a coupling cycle. sload() called with R1 = 20%, R2 = 60%, and R3 = 20% of the simulated time.

state and flux data.

To simulate constituents' computational load, including intraconstituent communication, we created sload(), which uses sleep() based on linear interpolation of CCSM3 benchmarking data [21]. In addition to sload(), constituents call mctload(), send(), and recv() in each time step. The mctload() function performs parallel data transformations, such as intergrid interpolation and time integration of flux and state data.

Interconstituent communications between cpl and other constituents are performed through the MCT send() and receive() communication calls. The ocn constituent runs with the time loop representing a day and communicate with cpl once. However, the other constituents run with cpl in a nested time loop representing 24 hours [22]. Note that cpl performs send and receive operations with land and ice simultaneously.

The total computation time of an iteration as obtained from the CCSM3 benchmarking data was divided into three different tasks, occupying 20%, 60%, and 20% of the benchmarked total iteration time, respectively. These were simulated by calling sload() at three different call sites in each simulated day and hour. The interconstituent communication patterns with three different amounts of load together results in complex interactions between constituents.

We ran the performance studies with two different PE cohort sizes. In Scenario 1, the benchmark ran with 208 PEs ($P^0 = 208$) so as to evaluate the LBM optimizer employing the automatically generated prediction model (AUTO) in the case where there are sufficient computational resources to reach the optimal processor allocation. For Scenario 2, we used 124 PEs ($P^0 = 124$) so as to run the benchmark with AUTO under an underprovisioned situation. In Scenario 3, we also allocated 124 PEs ($P^0 = 124$) to the benchmark. However, we used the LBM optimizer employing a manually determined heuristic prediction model (MANUAL) presented in our previous work [2] for the benchmark.

Because the coupling patten is significantly complex due to the complex load with differing amounts of computation in intervals between communications, it is onerous to determine a precise hand-tuned performance model. Thus, we assume cpl, atm, and land or cpl, atm, and ice perform most computation sequentially with each other, while ocn mostly

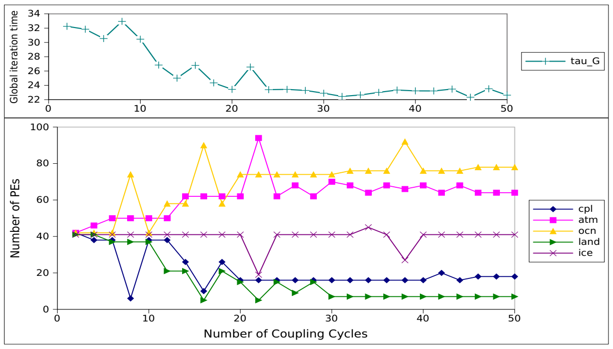


Fig. 5. Scenario 1: Optimization with AUTO and 208 PEs: The LBM found (18, 64, 78, 7, 41) with τ_G = 22.31 s at the 45th coupling cycle from INIT1 with initial τ_G = 32.25 s.

runs concurrently to the sequential execution for MANUAL so that we allowed MANUAL to use three static linear sums of unweighted τ_i^{comp} of constituents as τ_G estimators:

$$\begin{aligned}\tau_G = \max\{&\tau_{cpl}^{comp} + \tau_{atm}^{comp} + \tau_{land}^{comp},\\ &\tau_{cpl}^{comp} + \tau_{atm}^{comp} + \tau_{ice}^{comp},\\ &\tau_{ocn}^{comp}\}\end{aligned} \tag{12}$$

For all three scenarios, the LBM optimizer used the modified piecewise linear interpolation.

The *prediction counter* for the LBM optimizer was initially set $\widehat{\Delta P} = 2^3$ and increased or decreased geometrically by a factor of 2 but restricted to the range $2^1 \leq \widehat{\Delta P} \leq 2^5$.

Experiments using the benchmark were performed on a 32-node Linux cluster at SUNY Binghamton. Each node had dual 2.33 GHz Xeon quad-core processors with 8 GB memory. Nodes communicate via a Gigabit Ethernet interconnect. We distributed each constituent's PEs randomly across the cluster. We found that doing so helps prevent network congestion and packet loss, thereby minimizing timing variability. We used MPICH2 1.2.1 and MPI synchronous mode send to prevent unexpected timing behavior due to MPI buffering mechanism.

The experimental average results obtained by running the benchmark with LBM 10 times under Scenario 1, Scenario 2, and Scenario 3 are summarized in Table I. Values of τ_G are obtained by the LBM optimizer for five sets of initial conditions P^0. The number of coupling cycles required to find a PE allocation solution is presented with the number of reallocations (#REALLOC) and percentage of reallocations that were undone (UNDO(%)). Reallocation efficiency (Efficiency) is calculated by dividing τ_G value reduction by the number of reallocations. We use a coupled model PE configuration as $\vec{P} = (N_{\mathsf{cpl}}, N_{\mathsf{atm}}, N_{\mathsf{ocn}}, N_{\mathsf{land}}, N_{\mathsf{ice}})$ to describe PE allocation of the benchmark.

A. Scenarios 1: Optimization using AUTO with 208

We ran the benchmark with $P^0 = 208$ PEs with AUTO for Scenario 1 from five initial configurations: INIT1, where $\vec{P}^0 = (42, 42, 42, 42, 42)$; INIT2, where $\vec{P}^0 = (10, 118, 50, 10, 20)$; INIT3, where $\vec{P}^0 = (10, 40, 128, 10, 20)$; INIT4, where $\vec{P}^0 = (10, 40, 50, 88, 20)$; and INIT5, where $\vec{P}^0 = (10, 40, 50, 10, 98)$. We intended INIT1 to allocate

TABLE I
LBM CONVERGENCE PROPERTIES AND GLOBAL ITERATION TIME STATISTICS

	Scenario 1 (208 PEs, AUTO)					Scenario 2 (124 PEs, AUTO)					Scenario 3 (124 PEs, MANUAL)				
	INIT1	INIT2	INIT3	INIT4	INIT5	INIT1	INIT2	INIT3	INIT4	INIT5	INIT1	INIT2	INIT3	INIT4	INIT5
Initial tau_G	32.3	30.35	32.27	35.08	34.81	55.97	45.1	50.26	58.88	54.12	56.08	45.44	50.05	58.74	54.27
Final tau_G	22.95	22.95	22.47	23.05	23.43	34.85	34.72	34.57	34.55	34.84	35.1	36.23	50.22	36.59	36.11
Found at	37.6	53.4	53.4	58.4	45.6	35.6	50.8	53	46.2	37.2	93	126.4	0	122.6	135.4
#Realloc	18.3	26.2	26.2	28.7	22.3	17.3	24.9	26	22.6	18.1	46	62.7	0	60.8	67.2
Undo(%)	28.4	28.2	31.3	30.7	28.7	32.9	29.7	33.5	30.5	28.2	35.2	40.8	0	37	37.1
Efficiency	0.51	0.28	0.37	0.42	0.51	1.22	0.42	0.6	1.08	1.07	0.46	0.15	0	0.36	0.27

PEs between constituents uniformly. Moreover, INIT1 through INIT5 were intended to oversupply cpl, atm, ocn, land, and ice with PEs, respectively.

For reference purposes, we used the "ideal" value of τ_G = 22.07 s obtained by running the benchmark without MMCT and the LBM for $\vec{P} = (16, 64, 80, 16, 32)$ which is determined by CCSM developers via trial and error [21]. The five constituents of the benchmark repeatedly send timing data to the LBM, so that the LBM is allowed to detect and handle the load changes of constituents even though the optimization previously converged. Communication and synchronization overhead is manageable, currently adding 2.3% to the overall run-time. This is not a focus of the current work, and will be improved by future work.

Table I shows that the LBM optimizer with AUTO is able to find solutions for τ_G and corresponding PE allocations that are close to the ideal solution from all initial cases. For all initial cases, we obtained τ_G reduced by 30.1% through 23.3 reallocations at 47th coupling cycle on average. The LBM found the best solution for $\tau_G = 22.47$ s from INIT3 that ocn is initially oversupplied with PEs. Because ocn communicates only once with cpl within a coupling cycle, ocn's throughput does not affect τ_G significantly. This allows the LBM optimizer to simply predict τ_G improvement by taking PEs from ocn and allocating them to others until ocn's computation time is the bottleneck of τ_G. By comparison, the prediction is more difficult and more disturbed from timing noise if cpl, atm, land, or ice is allocated oversupplied PEs since their τ_i^{comp} changes influence τ_G much due to 24 interconstituent communications within a coupling cycle. It caused the LBM to find the worst average solution for $\tau_G = 23.43$ from INIT5.

LBM's convergence behavior of $\vec{P}$ and τ_G under Scenario 1 from INIT1 is well presented in Figure 5. The optimization was generally proceeded by donating PEs from cpl and land to atm and ocn. The LBM also performed three reallocations for ice at 21st, 33rd and 37th coupling cycles but these were undone immediately at the next LBI due to τ_G increment. As a result, the LBM found a solution for τ_G = 22.31 s almost identical to the ideal τ_G value with $\vec{P} = (18, 64, 78, 7, 41)$ through total 22 reallocations including 7 reallocations undone to recover previous PE allocation. Note that the automatic prediction model was generated at each LBI using various number of partial τ_i^{comp} each of which measured in an interval between synchronized points depending on τ_i^{comp} of constituents. For example, the model was created by 54, 3, 146 measurements at 1st, 2nd, and 3rd LBI respectively.

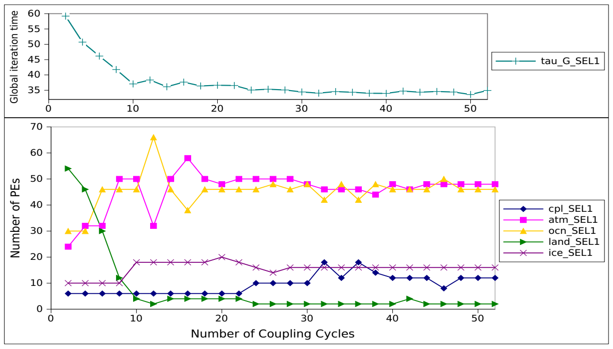


Fig. 6. Scenario 2: Optimization with AUTO and 124 PEs: The LBM found (12, 48, 46, 2, 16) with τ_G = 34.53 s at the 47th coupling cycle from INIT4 through 23 reallocations (Undo: 30.43%).

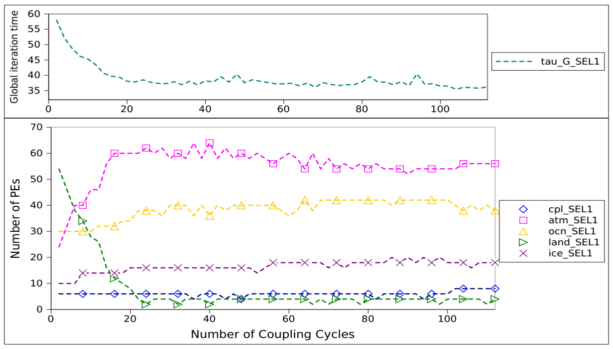


Fig. 7. Scenario 3: Optimization with MANUAL and 124 PEs: The LBM found (8, 56, 38, 4, 18) with τ_G = 36.18 s at the 111st coupling cycle from INIT4 through 55 reallocations (Undo: 34.55%).

B. Scenarios 2 and 3: Optimization using AUTO and MANUAL with 124 PEs

In Scenario 2 and 3 we ran the benchmark with a total of 124 PEs which is not enough to fulfill the optimal allocation. The initial configurations were: INIT1, where $\vec{P}^0 = (25, 25, 25, 25, 25)$; INIT2, where $\vec{P}^0 = (6, 72, 30, 6, 10)$; INIT3, where $\vec{P}^0 = (6, 24, 78, 6, 10)$; INIT4, where $\vec{P}^0 = (6, 24, 30, 54, 10)$; and INIT5, where $\vec{P}^0 = (6, 24, 30, 6, 58)$.

In Table I, it can be seen that the LBM with AUTO under Scenario 2 found a better solution for τ_G from all initial cases than did the LBM optimizer employed MANUAL in Scenario 3.

For Scenario 2, the LBM optimizer employing AUTO discovered PE allocation with overall τ_G = 34.71 s at 45th coupling cycle through 21.8 reallocations from all initial PE allocations on average, resulting in obtaining 33.8% reduced τ_G approximately. By comparison, for initial cases except

INIT3, the LBM optimizer with MANUAL found worse solution of $\tau_G = 36.01$ s with corresponding PE allocation overall. Furthermore, it required 59.2 reallocations that are significantly more than using AUTO on average under Scenario 3 since the heuristic prediction model that manually determined provided inaccurate τ_G prediction that resulted in inefficient and unnecessary reallocations which could be undone finally. It also caused less reallocation efficiency with MANUAL than AUTO. For INIT3, the LBM even never tried to reallocate PE allocation because the predicted τ_G through the inaccurate manual prediction model never estimated less τ_G value than the measured τ_G. This comparison result shows that the prediction accuracy influences the load-balancing performance significantly and AUTO is able to provide accurate τ_G prediction.

The convergence behaviors with AUTO and MANUAL through reallocations from INIT4 are shown in Figure 6 and Figure 7. The LBM with AUTO forced land hand redundant PEs over to atm, ocn, and ice initially. Once land finished donating 52 PEs at 11th coupling cycle through 5 reallocation, other constituents mainly reallocated their PEs appropriately. It resulted in the configuration with $\tau_G = 34.53$ s after 23 reallocations of which 30.43% was undone. With the LBM employing MANUAL land also kept donating 52 PEs to other constituents. However, this donation required 6 more reallocations to be completed at 23rd coupling cycle than the LBM with AUTO. Moreover, after 23rd coupling cycle, all constituents perform reallocations frequently. It caused that the LBM with MANUAL found the PE allocation with larger $\tau_G = 36.18$ s at 111th coupling cycle through 55 reallocations whose 34.55% was undone than the LBM with MANUAL due to inaccurate τ_G prediction.

VII. Conclusions and Future Work

Successful deployment of complex multiphysics and multiscale models on exascale supercomputers will require overcoming unique scalability challenges. A key capability these systems will require is generic, dynamic load-balancing infrastructure that can operate successfully in the presence of coupling-imputed serializations.

We have presented a technique for automatically generating performance predictions for coupled models from timing measurements and communication information. This technique can identify and imputed serializations, quantify their effects, and forecast coupled model throughput resulting from candidate interconstituent load balance reconfigurations.

Our experimental results show that, when used in our previously developed framework, our automatic performance models are effective and competitive with the best hand-tuned results, and outperforms the manually determined prediction model described in our previous work [2]. Future work will concentrate on reducing overhead, applying our performance model generation scheme to other test applications, and extending our analysis to include higher-order effects such as interconstituent $M \times N$ communications pattern costs and constituent scaling heterogeneity within the coupling cycle.

References

[1] D. G. Feitelson and L. Rudolph, "Towards convergence in job schedulers for parallel supercomputers," in *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*. Springer-Verlag, 1996, pp. 1–26.

[2] D. Kim, J. W. Larson, and K. Chiu, "Malleable model coupling with prediction," in *Proceedings of the 12th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID '12*, 2012, pp. 360–367.

[3] ——, "Dynamic load balancing for malleable model coupling," in *Proceeding of 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA '12*, 2012, pp. 150–157.

[4] "Community Climate System Model web site," http://www.cesm.ucar.edu/models/ccsm3.0/.

[5] S.-H. Ko, N. Kim, J. Kim, A. Thota, and S. Jha, "Efficient runtime environment for coupled multi-physics simulations: Dynamic resource allocation and load-balancing," in *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 349–358.

[6] S. S. Vadhiyar and J. J. Dongarra, "SRS - a framework for developing malleable and migratable parallel applications for distributed systems," *Parallel Processing Letters.*, vol. 13, no. 2, pp. 291–312, 2003.

[7] K. E. Maghraoui, B. K. Szymanski, and C. Varela, "An architecture for reconfigurable iterative MPI applications in dynamic environments," in *Proceedings of the Sixth International Conference on Parallel Processing and Applied Mathematics (PPAM2005), number 3911 in LNCS*. Springer Verlag, 2005, pp. 258–271.

[8] K. El Maghraoui, T. J. Desell, B. K. Szymanski, and C. A. Varela, "Dynamic malleability in iterative MPI applications," in *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid, CCGRID '07*. IEEE, 2007, pp. 591–598.

[9] R. Sudarsan and C. Ribbens, "ReSHAPE: A framework for dynamic resizing and scheduling of homogeneous applications in a parallel environment," in *Parallel Processing, 2007, ICPP2007*. IEEE, 2007.

[10] G. Utrera, J. Corbaln, J. Labarta, and D. D. D. Computadors, "Implementing malleability on MPI jobs," in *Proceedings of the 13th International Conference on Parallel Architecture and Compilation Techniques (PACT'04)*. IEEE Computer Society, 2004, pp. 215–224.

[11] "Institute for Combinatorial Scientific Computing and Petascale Simulations," http://www.cscapes.org/.

[12] U. Catalyurek, E. Boman, K. Devine, D. Bozdag, R. Heaphy, and L. Riesen, "Hypergraph-based dynamic load balancing for adaptive scientific computations," in *Procceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS'07)*. IEEE, 2007.

[13] C. Huang, G. Zheng, L. Kal, and S. Kumar, "Performance evaluation of adaptive mpi," in *Proceedings of the 11st ACM SIGPLAN symposium on Principles and practice of parallel programming*, ser. PPoPP '06. ACM, 2006.

[14] L. V. Kale and G. Zheng, "Charm++ and AMPI: Adaptive Runtime Strategies via Migratable Objects," pp. 265–282, 2009.

[15] S. G. Parker, "A component-based architecture for parallel multi-physics pde simulation," *Future Gener. Comput. Syst.*, vol. 22, no. 1, pp. 204–216, 2006.

[16] J. Luitjens and M. Berzins, "Improving the performance of uintah: A large-scale adaptive meshing computational framework." in *IPDPS*. IEEE, 2010.

[17] P. H. Worley, A. A. Mirin, A. P. Craig, M. A. Taylor, J. M. Dennis, and M. Vertenstein, "Performance of the community earth system model," in *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. ACM, 2011, pp. 54:1–54:11.

[18] D. Kim, J. W. Larson, and K. Chiu, "Toward malleable model coupling," *Procedia Computer Science*, vol. 4, pp. 312–321, 2011.

[19] J. Larson, R. Jacob, and E. Ong, "The Model Coupling Toolkit: A new Fortran90 toolkit for building multi-physics parallel coupled models," *Int. J. High Perf. Comp. App.*, vol. 19, no. 3, pp. 277–292, 2005.

[20] "Model Coupling Toolkit web site," http://mcs.anl.gov/mct/.

[21] J. W. Larson, R. L. Jacob, E. T. Ong, A. Craig, B. Kauffman, T. Bettge, Y. Yoshida, J. Ueno, H. Komatsu, S.Ichikawa, C. Chen, and P. Worley, "Benchmarking a parallel coupled model," *poster presented at Supercomputing '03*, 2003.

[22] J. W. Larson, "Ten organising principles for coupling in multiphysics and multiscale models," *ANZIAM Journal*, vol. 48, pp. C1090–C1111, 2009.