Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 831–840

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

Visualizing Process Composition and Load Balance in Parallel Coupled Models

J. Walter Larson[a,b,c]

[a] *Mathematics and Computer Science Division, Argonne National Laboratory*
[b] *Computation Institute, University of Chicago/Argonne National Laboratory*
[c] *Research School of Computer Science, The Australian National University*

Abstract

Coupled model development presents a set of challenges broadly called the coupling problem; message-passing parallelism complicates matters, resulting in the parallel coupling problem. Performance tuning of parallel coupled systems is complex and performed largely in an ad hoc fashion; from the domain scientist's perspective the figure of merit is throughput, which is the amount of simulation achieved per unit of wall-clock time. Achieving high throughput for parallel coupled models requires high scalability for each subsystem and compatible combinations of the subsystems' respective resource allocations (e.g., MPI processes) to minimize idle time surrounding coupling events. Scaling parallel coupled models up to million-way parallelism highlights the need for practical methods for describing and evaluating these systems. I present a set of complementary tools to analyze and visualize process composition and load balance for coupled models. I state five basic process compositions found in coupled models. Two are the irreducible, well-known sequential and parallel compositions found in common process algebras. I define three new derived process compositions that appear in coupled systems. I define a dynamic load balance hierarchy. I propose a simple graph-based schema for diagramming process composition in coupled models that is capable of expressing dynamic load balance relationships, and I present simple examples illustrating its use. I apply the graphical schema to Version 4 of the Community Climate System Model to estimate the complexity of the process composition and load balance problem for this system.

Keywords:
Multiphysics Modeling; Multiscale Modeling; Process Composition; Load Balance; Computational Complexity

1. Introduction

The prefix "multi" in multiphysics and multiscale models signifies that these systems are composites of multiple interdependent subsystem models. These inter-subsystem data dependencies, or *couplings*, pose a computational science problem called the *coupling problem* [1]: the description, transmission, and transformation of output from one subsystem into input to another. Examples of description include fields under exchange and the domains on which they reside. Transmission is merely the hand-off of data from one subsystem to another. Transformation is the set of computations applied to a source subsystem's output that results in input data for a target subsystem (e.g., intergrid interpolation). On a platform possessing a single address space, data transmission is straightforward (e.g., arguments supplied to a function call or Fortran common blocks), as are data description and transformation. Many coupled

Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.088

systems, however, owe their existence to the computational power available through parallel computing and, in particular, message-passing parallelism employing the Message Passing Interface[1] (MPI) Standard [2]. Message passing and distributed memory transmogrify the coupling problem into a much harder problem—the *parallel coupling problem* (PCP) [1]. The PCP comprises distributed data description, parallel data transmission (the $M \times N$ problem [3]), and parallel data transformation. Introducing concurrency adds design degrees of freedom, most notably process composition, implementation in single or multiple executables, and load balance. This paper focuses on process composition and its relationship to load balance in parallel coupled systems; other aspects of the PCP are beyond the scope of this work.

Performance tuning of parallel coupled models is a complex problem; to date it has largely been performed in an ad hoc fashion. From the domain scientist's perspective the most vital performance characteristic for a parallel coupled model is *throughput*, that is, the amount of simulation achieved per unit of wall-clock time (e.g., model-years per wall-clock day for a climate model). Achieving high throughput for these models requires high scalability for each subsystem and compatibility among the subsystems' respective resource allocations (i.e., MPI processes) to minimize idle time surrounding coupling events. High-performance computing is moving rapidly toward offering up to million-way parallelism; utilizing this power will require coupled models to undergo unprecedented levels of performance analysis and optimization. Clearly needed are simple and practical methods for description, analysis, and visualization of these models.

In this paper, I offer a set of definitions and concepts for describing two intertwined architectural degrees of freedom in coupled systems: process composition and load balance. I state three basic process compositions found in coupled models. Two are the irreducible, well-known sequential and parallel compositions found in process algebras. I propose three new derived process compositions that appear in coupled systems. I formalize the well-known notion of the PE versus time graph, using it to illustrate the aforementioned process compositions, and I remark on its relationship to load balance optimization of coupled models. I define a hierarchy of dynamic load balance categories and discuss the set of requirements each imposes on a coupled system.I propose a new graphical schema for diagramming process composition in coupled models and present simple illustrative examples. I apply the graphical schema to estimate the complexity of the load balance tuning problem for Version 4.0 of the Community Climate System Model (CCSM4), and relate this to the results of other researchers' performance tuning of this model.

2. Process Composition in the Parallel Coupling Problem

Defined below is a set of basic terms for describing parallel coupled systems. Some concepts were first defined in [1], while others are extensions thereof specific to process composition and load balance.

2.1. Basic Definitions

A *coupled model* consists of a set of N constituent models, or *constituents*[2] $\{C_1, \ldots, C_N\}$. Two constituents C_i and C_j are *coupled* if they either share at least one explicit input/output dependency or require implicit simultaneous self-consistent solution of their state due to shared variables—*explicit* and *implicit* coupling, respectively. The coupled system is laid out across a global set S of P *processing elements* (PEs). Each constituent C_i executes on a set s_i of $p_i \leq P$ PEs called a *cohort*. In set notation, $|S| = P$, $|s_i| = p_i$, $s_i \subseteq S$, and $S = \bigcup_{i=1}^{N} s_i$

Time integration of a parallel coupled model proceeds as follows. Individual constituents solve their respective equations of evolution using (providing) input (output) from (to) other constituents; these calculations are performed on the constituent's PE cohort. Each constituent C_i has its timestep Δt_i, which may or may not be constant. A constituent C_i interacts with other constituents during *coupling events*; these events involve communication between constituents' PE cohorts and, in some cases, calculations performed on the union of their cohorts. Coupling events

[1] MPI has attained the status of an "industry standard" for scientific high-performance computing, and from this point forward my use of the word "parallel" will amount to MPI-based parallelism.

[2] The term *constituent* was first defined in [1]. Traditionally, coupled model developers have used the term *component model* or *component* instead. I introduced the term "constituent" to avoid confusion with the term "component" from component-based software engineering—a technique sometimes employed in building coupled models.

either are state-threshold-driven and potentially minimally predictable (or even unpredictable) or *scheduled* (i.e., coupling times are known a priori). In many coupled systems, coupling events are all scheduled; these scheduled events may be mutually commensurate and thus fall within a repeatable *coupling cycle*.

For example, consider a coupled climate model. The timestep for atmosphere and ocean models is typically on the order of minutes; coupling events are scheduled hourly and fall within a coupling cycle of one model day. Thus time-to-solution measurements center on the length of the coupling cycle, leading to throughput defined in terms of model days or years per unit of wall-clock time.

2.2. Process Composition in Parallel Coupled Models

The assignment of constituents to their respective cohorts is called *process composition*. Two irreducible process compositions exist, *sequential* and *parallel* [4]. In a sequential composition, $s_1 = \cdots = s_N = S$, and the constituents C_i execute in turn as an event loop on the global cohort. In a parallel composition, $s_i \cap s_j = \emptyset$ for $i \neq j$. Two other types of process compositions have been found in coupled models [1], *overlapping* and *nested*.[3] In an overlapping composition, each constituent cohort shares at least one PE with another constituent, but the overlapping cohorts are not identical; in an overlapping composition having N constituents, $\forall i \in \{1, \ldots, N\}\ \exists j \in \{1, \ldots, N\}$ with $j \neq i$ such that $s_i \cap s_j \neq \emptyset$ and $s_i \neq s_j$. Overlapping compositions are suitable to an implicit coupling problem, for example core-edge coupling in fusion plasmas [5]. A nested composition combines two or more irreducible composition operations, with at least two of the types (sequential, parallel, overlapping) present.

The composition strategies discussed thus far have been assumed *static*; in principle it is possible to apply the previously stated process compositions on the fly, but this dynamism is not a consequence of their definitions. A process composition that is inherently dynamic is the *rendezvous*, under which two or more processes in parallel composition synchronize and then execute operations on the union of their cohorts. Dynamic interconstituent load balance—removing PEs from one constituent and assigning them to another—is another example of nonstatic process composition (see Section 2.3 for further discussion).

Process composition can be represented by using a process calculus [6]. Process calculi, though useful for detailed analysis of concurrency and communications in large systems, present the non-computer scientist a steep learning curve and may only offer binary composition operators where n-ary ones are needed to describe most coupled models. I propose a simpler set of tools for *visualizing* process composition so that coupled model developers can understand at a glance what a code's PE layout is and how constituents are mapped to this layout and scheduled for execution. In Section 2.4 I formalize the PE versus time plot, and in Section 3 I propose a new graph-based schema for analyzing process composition and load balancing.

2.3. Load Balance in Parallel Coupled Models

Load balance in a parallel coupled model is the assignment of sizes to constituent PE cohorts. The object is to choose cohort sizes that are well balanced from the standpoint of minimizing the global integral of CPU time wasted through load intraconstituent load imbalance or idle time surrounding coupling events. Load-balancing strategies for coupled models can be classified according to how processors are allocated to constituent subsystems and whether these assignments are static or dynamic. Furthermore, these strategies may be characterized according to the scope over which they are conservative—within a constituent's cohort s_i, within the global cohort S, or not at all. *Level 0 dynamic load balance* is not at all dynamic; it is static resource allocation in which the size of the global processor pool is fixed, each constituent is assigned a fixed set of processors, and the decomposition of tasks among processors does not vary in time. *Level 1 dynamic load balance* allows dynamic intraconstituent dynamic load balance but does not allow reassignment of processors from one constituent to another. *Level 2 dynamic load balance* allows interconstituent dynamic load balance under which processors may be taken from one constituent's cohort and assigned to another, with the constraint of a fixed global processor pool. *Level 3 dynamic load balance* allows the size of the global processor pool to be expanded or contracted dynamically in addition to any interconstituent dynamic load balancing that may occur. Table 1 summarizes these strategies.

Software requirements for implementing a given level of load balance are cumulative. Level 0 dynamic load balance is purely static and is the easiest to implement. Level 1 dynamic load balance for a given constituent requires

[3] In [1] I used the term *hybrid* rather than *nested*; the latter is preferable because it is more descriptive.

Table 1: Levels of Dynamic Load Balance

Level	Within Constituent	Constituent Cohort s_i	Global Cohort S	Conservative w.r.t. s_i?	Conservative w.r.t. S?
0	Static	Static	Static	Yes	Yes
1	Dynamic	Static	Static	Yes	Yes
2	Dynamic	Dynamic	Static	No	Yes
3	Dynamic	Dynamic	Dynamic	No	No

checkpointing of its internal state, migration of state data from the old layout to the new one, and re-handshaking of $M \times N$ connections to other constituents. Level 2 dynamic load balance additionally requires a coupling infrastructure capable of resizing PE cohorts (e.g., creating new MPI communicators) and the ability to migrate, instantiate, and initialize a constituent on each PE added to its associated cohort. Level 3 dynamic load balance additionally requires the coupling infrastructure to be able to grow or shrink the global processor pool; for an MPI application, this amounts to a dynamic MPI_COMM_WORLD. Though the requirements for Level 2 and Level 3 dynamic load balance appear demanding, work is under way to meet them in a widely used coupling package [7]. This effort will implement flexibility in cohort and global PE pool sizes, a property called *malleability*. Level 2 dynamic load balance gives rise to a process composition strategy called *balancing*. Level 3 requires *expansion* and *contraction* composition strategies.

2.4. PE versus Time Plot

A simple way to visualize process composition in coupled systems is a *PE versus time* (PVT) plot (Figure 1). The vertical and horizontal axes in a PVT plot are time and PE rank on the system's global communicator; thus length in the horizontal dimension is incremented in PEs[4], and area on the plot has units in PE·sec. Note the time axis has an elliptical arrow adjacent to it, indicating the assumption for this discussion that the coupled model has scheduled coupling, and the set of all the interconstituent couplings falls into a coupling cycle of constant period; this assumption is valid for a large variety of coupled systems—for example, climate and Earth system models. A sequential composition (Figure 1(a)) is a set of horizontal bands, each color representing a different constituent. A parallel compsition (Figure 1(b)) appears as a set of vertical bands colored to indicate each constituent. An overlapping composition (Figure 1(c)) appears as vertical bands that overlap; here C_1 and C_2 occupy PEs 0–4 and 3–7, respectively, with PEs 3–4 shared. Two simple nested compositions, sequential over parallel and parallel over sequential, are shown in Figures 1(d) and 1(e), respectively. In Figure 1(d) a sequential composition schedules C_1 to execute on the global cohort S; next, the global cohort is split between C_2 and C_3, which execute concurrently on their respective disjoint subsets of S; finally, C_4 executes on the global cohort S. In Figure 1(e) C_1 and C_4 execute continuously on cohorts $s_1 = \{0, 1\}$ and $s_4 = \{6, 7\}$, respectively, while C_2 and C_3 execute in turn on $s_2 = s_3 = S - (s_1 \cup s_4) = \{2, 3, 4, 5\}$.

PVT plots have appeared informally in the coupled model literature (e.g., Figure 1 in [8]), and the notion of coloring such plots based on code profiling information forms the basis of performance visualization tools such as Jumpshot [9]. The PVT plots provide an intuitive approach to multimodel load balance as follows:

introducing higher levels of granularity in PVT plots allows identification of key performance thieves such as load imbalance and idling awaiting interconstituent data transmission; optimization amounts to *minimization* of the total area on the PVT plot occupied by these operations (cf. Figure 7 of [10]). Thus PVT plots can guide decisions on which of a set of candidate process compositions and resource allocations best fits the model. Drawing PVT plots is relatively easy, but entwining the dimension of PEs makes this approach harder to scale hand-drawn diagrams to large numbers of constituents. Furthermore, as described here, PVT plots are not rich enough to describe rendezvous and load-balancing compsition operators.

[4]The discussion here assumes a homogeneous cluster, but can be extended to describe heterogeneous systems by varying the widths of the PE increments corresponding to PE capability (e.g., peak floating-point operations rate).

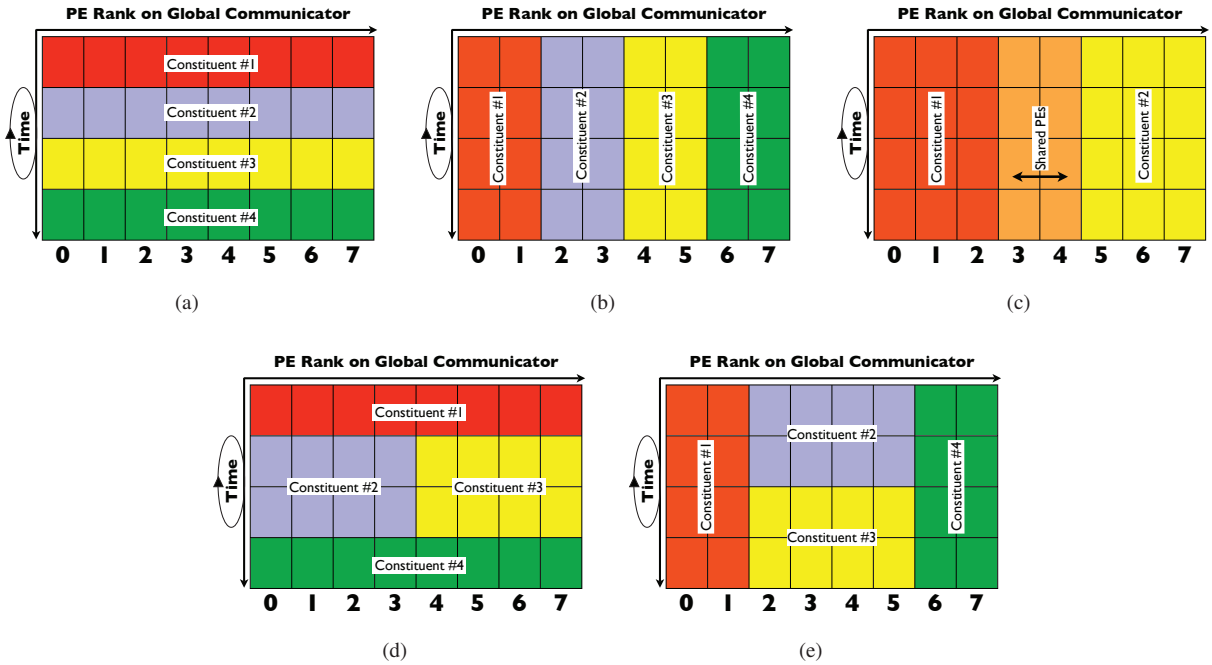


Figure 1: PE versus time plots for: (a) sequential composition, (b) parallel composition, (c) overlapping composition, (d) sequential-over-parallel nested composition, and (e) parallel-over-sequential nested composition.

3. Graphical Schema for Representing Process Composition

Below I define a graph-based schema for representing process compsition and load balance in parallel coupled models.

3.1. Schema Definition

One approach to representing process composition is a directed *graph* [11] in which the vertices correspond to composition strategies and constituents, and directed edges connect them to represent compositional relationships. The resulting graph $\mathcal{Q}$ is the *process composition graph* (PCG) for the coupled model $\mathcal{M}$ it represents.

Figure 2 shows the core vertex symbol set and vertex-edge usage. The generic vertex symbols are defined in Figure 2(a); in the text I will use sans-serif letters corresponding to those on the symbols to refer to a vertex on a given graph; for example S for sequential composition, P for parallel composition, et cetera. Symbols may be annotated (Figure 2(b)) to provide additional information regarding how they are implemented in $\mathcal{M}$—for example Px, Sf, and Cx denoting parallel compsition implemented as an executable driver, serial compsition implemented as a function within an executable, and a constituent implemented as a stand-alone executable image, respectively.

The rules for constructing a PCG are:

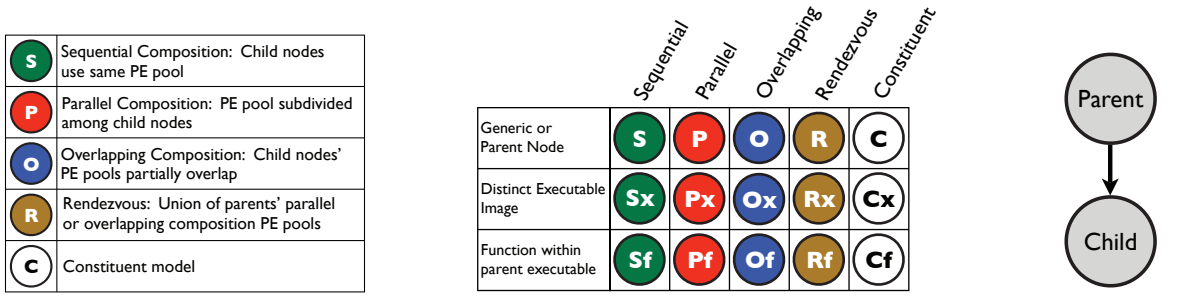
R1 Only one edge may connect a parent vertex to a child vertex (Figure 2(c)).

R2 Each constituent C_i must appear somewhere on $\mathcal{Q}$ as a vertex that is the child of a composition vertex.

R3 Each constituent vertex must have only one parent, and this parent must be a composition vertex.

R4 Any composition vertex of type S, P, or O must have at a minimum two edges directed away from it toward child vertices.

R5 No composition vertex may have as its child a composition vertex of the same type.



(a) Fundamental process composition node symbols.

(b) Specialized process composition node symbols.

(c) Parent-child connection.

Figure 2: Vertex types and vertex-edge usage.

R6 The rendezvous R vertex must have in-valency greater than or equal to two; that is, it must have at least two *parents*, and all its parents must be constituent, rather than composition, vertices.

R7 Only constituent vertices may have zero out-valency; these vertices are called *leaves*.

The PCG Q is *connected*; every node positive in- or out-valency. If a PCG has no rendezvous R or load balance B (see Section 3.3) vertices, then it is *acyclic* and represents a static process composition and load balance configuration. Furthermore, in the absence of R and B vertices, the PCG is a *tree*; the maximum number of children k of all of the PCG's vertices makes it a *k-ary tree*.

3.2. Simple Examples

Below I present simple illustrative examples to demonstrate usage of the schema defined in Section 3.1.

Climate models have long been parallel coupled models; the Parallel Climate Model (PCM) and CCSM are typical examples. PCM employs a serial composition in a single executable (Figure 3(a))[12]. CCSM versions 1.0–3.0 employed a parallel composition implemented in multiple executables (Figure 3(b)) [13].

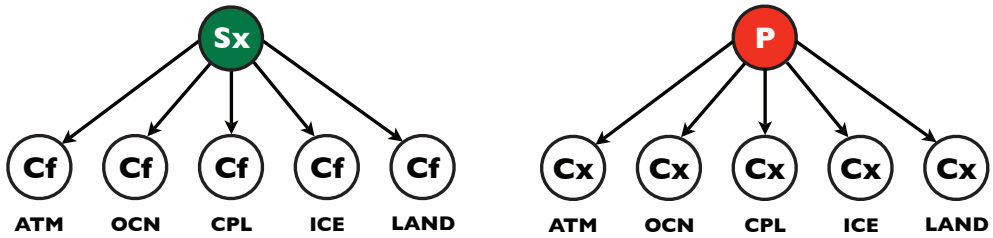
The Goddard Earth Observing System Data Assimilation System (GEOS-DAS) version 3.0 (Figure 4(a)) comprised three constituents, each a distinct executable image: a global forecast model; online observational quality control and model-analysis interfaces colloquially called the *plug*; and the Physical-space Statistical Analysis System (PSAS). Coupling was file-based. This system performed 3DVAR data assimilation: the forecast model produced a "background" forecast; the online observational quality control evaluated data from the observing system, eliminating bad data, while the plug interpolated the background forecast to the observation locations; the PSAS performed an analysis on the observational and background forecast data, producing an *analysis update* on the forecast model's grid to correct the background over the analysis period; the forecast model subsequently ran in assimilation mode, incorporating the analysis update incrementally.

Framework Application for Core-Edge Transport Simulations (FACETS) [5] is a system for coupling simulations of core and edge regions in fusion plasmas; the coupling is implicit because both core and edge models solve the same state equations (i.e., MHD). An overlapping composition is used to allocate core and edge PE cohorts; the shared PEs are used to run a nonlinear solver to arrive at a self-consistent, simultaneous state solution in the domain boundary region between core and edge (Figure 4(b)).

$M \times N$ transfers on intercommunicators (Figure 4(c)) can be viewed as a rendezvous process composition if the constituents executing the transfer delegate its execution to a separate constituent.

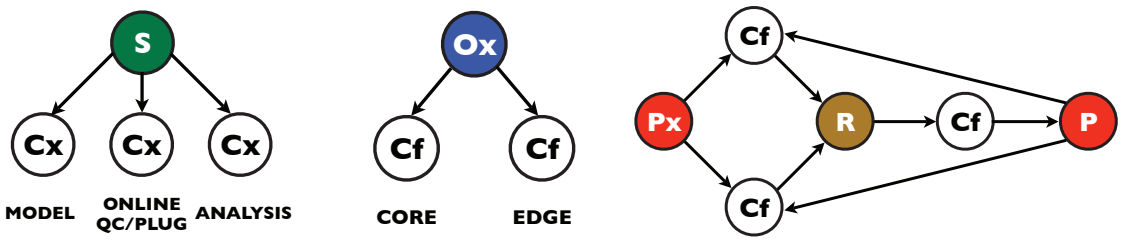
3.3. Representation of Dynamic Load Balance

The schema rules from Section 3.1 may be extended to support dynamic load balance operations. The PCG Q is no longer a digraph, but instead a *bidirected graph* [14]. Each edge in a bidirected graph has arrows at each end, pointing either away or toward the vertex it touches: *directed edges* have an arrow at each end pointing in the same direction and are frequently represented with one arrow pointing in that direction (just as in all the PCGs referred to



(a) Purely sequential composition in PCM. (b) Purely parallel composition in CCSM3.

Figure 3: Simple process composition in climate models.



(a) (b) (c)

Figure 4: Process composition in: (a) GEOS-3 data assimilation system; (b) core-edge coupling in FACETS; and (c) $M \times N$ transfer on an intercommunicator.

thus far in this paper); *extroverted edges* have arrows pointing outward at each end; and *introverted edges* have arrows pointing inward at each end. A new set of balance vertex symbols, B, +, and −, indicate generic load balance, and Level 3 expansion and contraction, respectively. The following extensions/amendments to **R1**–**R7** are necessary to describe levels 1–3 dynamic load balance from Table 1:

EXT1 The PCG Q is a bidirected graph with no introverted edges.

EXT2 The balance process composition operator B may be connected with a extroverted edge to a single parent constituent vertex to indicate intraconstituent (Level 1) dynamic load balance (Figure 5(a)).

EXT3 The balance operator B may be connected to multiple constituent parents to indicate Level 2 dynamic load balance on the union of their cohorts (Figure 5(b)).

EXT4 If the balance operator B is the child of a set of constituents that are collectively all the children of a parallel or overlapping composition parent vertex v^*, directed edges pointing into B and a single directed edge pointing out of B toward v^* indicates a level 2 dynamic balancing activity on the parent compsition v^* (Figure 5(c)).

EXT5 The balance operator B may be modified by the having either a single expansion (+) or contraction (−) operator as a parent to denote Level 3 expansion (Figure 5(d)) or contraction (Figure 5(e)), respectively.

4. Process Composition in CCSM4

Version 4.0 of CCSM (CCSM4) is a coupled climate model used by an international community of hundreds of scientists. CCSM4 is used to perform numerical studies of climate change, sensitivity, and variability and of paleoclimates. CCSM4 simulation results will be used in the upcoming Intergovernmental Panel on Climate Change's fifth scientific assessment report. CCSM4 comprises five constituents: atmosphere (ATM), ocean (OCN), land-surface (LAND), and sea-ice (ICE) models and a *coupler* (CPL). CCSM4 has a hub-and-spokes architecture; all coupling

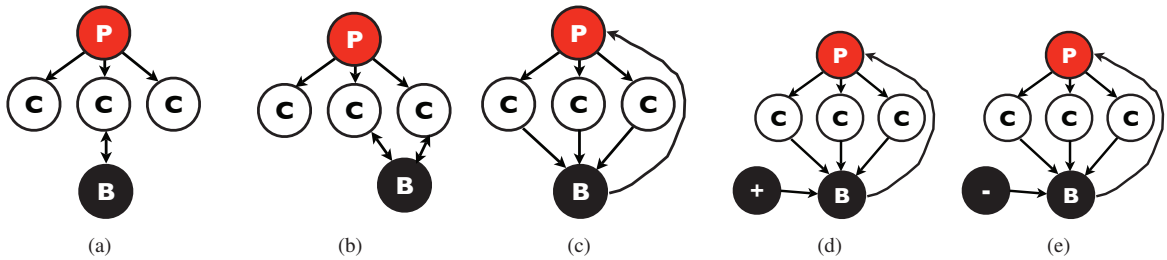


Figure 5: Graphical representation of dynamic load balance: (a) Level 1; (b) Level 2 bipartite; (c) Level 2 global; (d) Level 3 with expansion; and (e) Level 3 with contraction.

data traffic from/to the models (the spokes) is routed via the coupler (the hub), which performs regridding and flux calculations. The first three versions of CCSM employed a parallel composition with each constituent a separate executable (Figure 3(b)). CCSM4, however, is a single executable image; and its highly flexible coupling infrastructure CPL7 [10] allows sequential, parallel, and nested process composition strategies.

Using the methodology outlined in Section 3, I now demonstrate how to enumerate all of the possible compositions that employ sequential and parallel compositions and nestings thereof, and I show structural archetypes for each.

There is one purely sequential 3(b) and one purely parallel composition (not shown—simply replace vertices P and Cx in Figure 3(b) with Px and Cf, respectively). Next, consider single-level nesting of sequential over parallel compositions. According to the rules outlined in Section 3, there exist five archetypal structures for the compositions; Figure 6 shows them for sequential-at-the-top (SATT) nesting. From basic combinatorics, we can compute how many different combinations in which the five constituents can be mapped to each structure (this, of course, ignores the issue of how many PEs will be assigned at the PE cohort at each vertex). Archetypes SATT1A (Figure 6(a)) SATT1B (Figure 6(b))each offer $\binom{5}{3} = 10$ possibilities. SATT1C (Figure 6(c)) offers $\binom{5}{4} = 5$ combinations. SATT1D (Figure 6(d)) has $\binom{5}{3} = 10$ possibilities. SATT1E (Figure 6(e)) is slightly more complex: the number of possibilities is $\binom{5}{2}\binom{3}{2} = 30$. Thus, there are 65 possible process compositions for singly nested SATT. The singly nested parallel-at-the-top (PATT) archetypes result from interchanging S and P in Figure 6(a), yielding 65 possible process compositions. There exist five doubly nested SATT archetypes (Figures 7(a–e)). Archetypes SATT2A-D (Figures 7(a–d), respectively) each admit $\binom{5}{3}\binom{3}{2} = 30$ combinations. SATT2E (Figure 7(e)) admits $\binom{5}{3}\binom{2}{1} = 20$ combinations. Thus, there are 140 possible process compositions for doubly nested SATT; by symmetry, there are also 140 possible doubly nested PATT process compositions. Triply nested process compositions are possible with five constituents, and only one structural archetype exists for SATT (Figure 7(f)). This archetype admits $\binom{5}{2}\binom{3}{1}\binom{2}{1} = 60$ different process configurations; by symmetry, there also exist 60 triply nested PATT configurations. In sum, there are 2 pure, unnested compositions, 130 singly nested compositions, 280 doubly nested compositions, and 60 triply nested compositions, yielding a total of 472 process composition choices. Note that this analysis has considered only how each block is stacked, not their sizes (i.e., the cohort sizes $\{s_1, \ldots, s_N\}$).

Process composition / load balance combinations can show considerable variance in throughput. Craig et al. [10] benchmarked CCSM4 on 128 PEs of an IBM SP-6 for a moderate resolution configuration of 2° atmosphere and land grids combined with 1° ocean and sea-ice grids. Timings measured for a single model day at this resolution were 20.8 s for a purely sequential composition, 33.5 s for a purely parallel composition, 21.8 s for a PATT1C composition (Figure 8(a)), and 19.1 s for a PATT2B composition (Figure 8(b)). Their results are remarkable given how many possible process composition archetypes exist, combined with the combinatorics of PE cohort allocation. The wide variation in their results imply that a structured, systematic search algorithm that leverages the graphical schema would be useful in searching for process composition and load balance "sweet spots."

5. Conclusions and Future Work

Increasing computational capacity, better programming models, and the constant opening of new interdisciplinary fields of study guarantee the emergence of new multiphysics and multiscale models. Many coupled systems are developed for parallel platforms, and the rapid growth in parallelism on offer poses daunting performance engineering

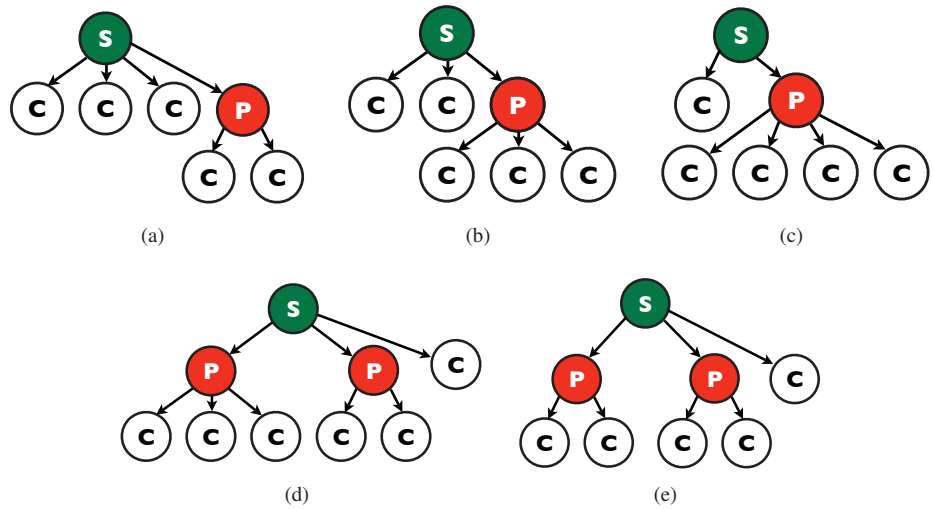


Figure 6: Singly nested SATT process composition structural archetypes: (a) SATT1A, (b) SATT1B, (c) SATT1C, (d) SATT1D, and (e) SATT1E.

challenges in terms of process composition and load balance. Conceptual frameworks and means for visualizing these challenges are needed to help formulate their solutions.

Conceptual frameworks for process composition and load balance in coupled models have been presented. The set of process compositions include new ones previously absent from common process calculi. The dynamical load balance hierarchy frames the discussion of runtime resource allocation for coupled models. Two graphical methods for analyzing parallel coupled systems have been presented: formalization of previously employed PVT plots and a new, graph-based methodology for enumerating process compositions. The graphical schema was applied to a case study of CCSM4 and elucidated the myriad process composition configuration choices available. The graphical schema is at the very least a useful bookkeeping tool for identifying potential model configurations.

At present, the graphical schema does not include *coupling* in its semantics. A set of extensions applying the *connectivity graph* [1] for a coupled system should be possible and will be a topic of future work. Another promising avenue of research will be combining the techniques presented here with detailed, comprehensive constituent benchmarking to form the basis for a coupled model throughput simulation system.

Acknowledgments

I thank Tony Craig for sharing the prepublication performance data for CCSM4 I quoted in this paper. This work was supported by the U.S. Department of Energy, under Contract DE-AC02-06CH11357.

References

[1] J. W. Larson, Ten organising principles for coupling in multiphysics and multiscale models, ANZIAM Journal 48 (2009) C1090–C1111.
[2] The Message Passing Interface (MPI) standard, `http://www-unix.mcs.anl.gov/mpi/`.
[3] F. Bertrand, R. Bramley, D. E. Bernholdt, J. A. Kohl, A. Sussman, J. W. Larson, K. Damevski, Data redistribution and remote method invocation for coupled components, J. Parallel Distrib. Comput. 66 (7) (2006) 931–946.
[4] I. Foster, Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering, Addison Wesley, Reading, Massachusetts, 1995.
[5] J. R. Cary, J. Candy, R. H. Cohen, S. Krasheninnikov, D. C. McCune, D. J. Estep, J. Larson, A. D. Malony, P. H. Worley, J. A. Carlsson, A. H. Hakim, P. Hamill, S. Kruger, S. Muzsala, A. Pletzer, S. Shasharina, D. Wade-Stein, N. Wang, L. McInnes, T. Wildey, T. Casper, L. Diachin, T. Epperly, T. D. Rognlien, M. R. Fahey, J. A. Kuehn, A. Morris, S. Shende, E. Feibush, G. W. Hammett, K. Indireshkumar, C. Ludescher, L. Randerson, D. Stotler, A. Y. Pigarov, P. Bonoli, C. S. Chang, D. A. D'Ippolito, P. Colella, D. E. Keyes, R. Bramley, J. R. Myra, Introducing facets, the framework application for core-edge transport simulations, Journal of Physics Conference Series 78 (2007) 0120086.
[6] M. Hennessy, Algebraic Theory of Processes, The MIT Press, Cambridge, Mass., 1988.
[7] D.-H. Kim, J. W. Larson, K. Chiu, Toward malleable model coupling, Preprint ANL/MCS-P1738-0310, Mathematics and Computer Science Division, Argonne National Laboratory (2011).

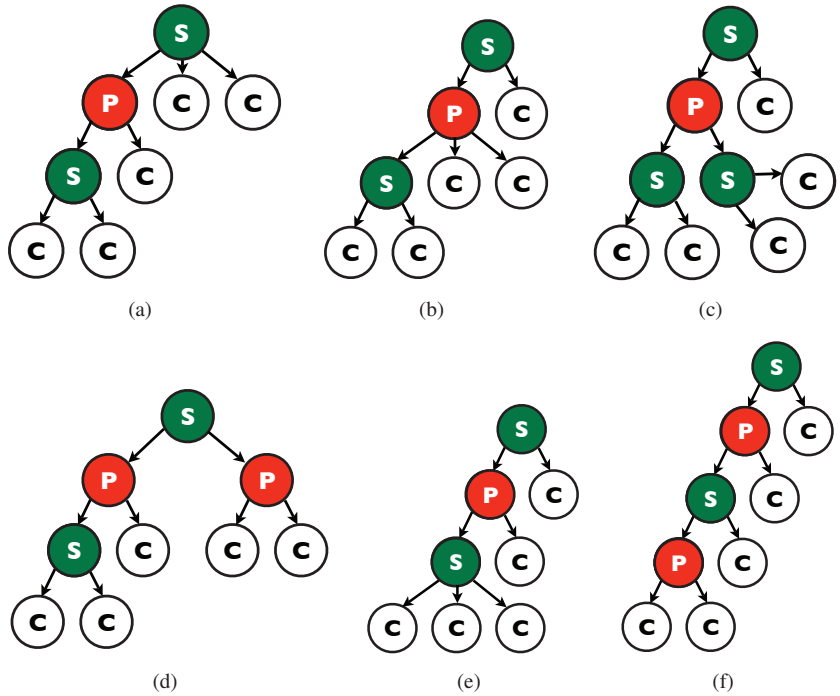


Figure 7: Doubly and triply-nested SATT archetypes: (a) SATT2A, (b) SATT2B, (c) SATT2C, (d) SATT2D, (e) SATT2E, and (f) SATT3.

[8] J. Larson, R. Jacob, E. Ong, The model coupling toolkit: A new fortran90 toolkit for building multi-physics parallel coupled models, Int. J. High Perf. Comp. App. 19 (3) (2005) 277–292. doi:10.1177/1094342005056115.
[9] Jumpshot users' guide, `http://www.mcs.anl.gov/research/projects/perfvis/software/viewers/jumpshot-4/usersguide.ht`
[10] A. P. Craig, M. Vertenstein, R. L. Jacob, A new flexible coupler for earth system modeling developed for CCSM4 and CESM1, submitted, International Journal of High Performance Computing Applications.
[11] R. Diestel, Graph Theory, 3rd Edition, Springer, New York, 2006.
[12] T. Bettge, A. Craig, R. James, V. Wayland, G. Strand, The DOE Parallel Climate Model (PCM): The Computational Highway and Backroads, in: V. N. Alexandrov, J. J. Dongarra, C. J. K. Tan (Eds.), Proc. International Conference on Computational Science (ICCS) 2001, Vol. 2073 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2001, pp. 148–156.
[13] A. P. Craig, B. Kaufmann, R. Jacob, T. Bettge, J. Larson, E. Ong, C. Ding, H. He, cpl6: The new extensible high-performance parallel coupler for the community climate system model, Int. J. High Perf. Comp. App. 19 (3) (2005) 309–327. doi:10.1177/1094342005056117.
[14] J. Edmonds, E. Johnson, Matching: A well-solved class of integer linear programs, in: M. Jnger, G. Reinelt, G. Rinaldi (Eds.), Combinatorial Optimization Eureka, You Shrink!, Vol. 2570 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2003, pp. 27–30.

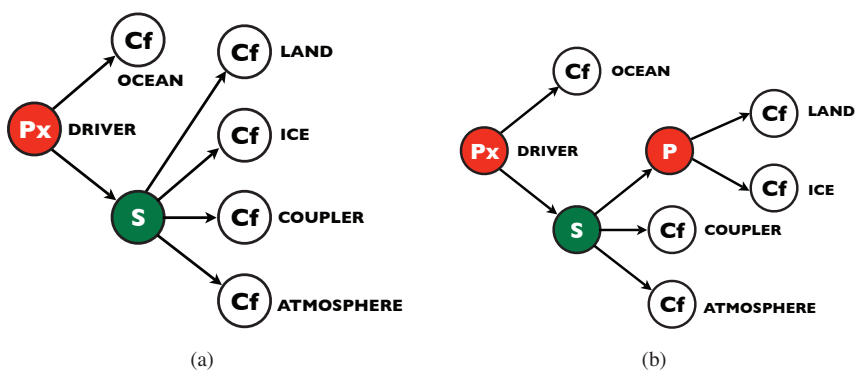


Figure 8: Nested process compositions for CCSM4: (a) singly nested and (b) doubly nested.