Analyzing the Impact of Computational Heterogeneity on Runtime Performance of Parallel Scientific Components

Sumir Chandra, Manish Parashar
Dept. of Electrical & Computer Engineering
Rutgers University, NJ, USA
Email: {sumir, parashar}@caip.rutgers.edu

Jaideep Ray
Advanced Software R & D
Sandia National Laboratories, CA, USA
Email: jairay@somnet.sandia.gov

Keywords: High performance computing, load balancing, component-based combustion applications, structured grids

Abstract

Scientific simulations modeling complex physical phenomena exhibit varying degrees of spatiotemporal and computational heterogeneity, which can pose significant challenges in algorithmic efficiency and runtime performance. Addressing these challenges often requires an understanding of application behavior and the impact of heterogeneity on the simulation, especially when analytical approaches are not feasible. Runtime calibration is used to analyze the impact of computational heterogeneity on the performance of two different load balancing strategies applied to two different problems of 2-D methane-air combustion using different chemistry models. Experimental evaluation demonstrates that such empirical approaches are inevitable in component-based scientific computations, where performance is determined by the problem characteristics and the particular connectivity of components in the simulation code, none of which are known before runtime.

1 INTRODUCTION

Numerical simulations offer the potential for accurate solutions of realistic models of complex time-evolving physical phenomena, such as combustion, and are playing an increasingly important role in science and engineering. The dynamic interactions underlying these physical phenomena span several different scales, resulting in significant spatiotemporal complexity. Moreover, these applications can change their mathematical and topological characteristics during the simulation and exhibit varying degrees of computational heterogeneity. In the context of this research, computational heterogeneity refers to the different computational requirements at each point in the simulation domain, which are typically manifested as pointwise varying workloads.

The inherent dynamism coupled with the runtime heterogeneity in scientific simulations lead to significant challenges in ensuring algorithmic efficiency, load balancing, and runtime performance management [1]. In cases where a formal analytical framework exists, e.g., error estimation via Richardson extrapolation [2] or the solution of linear systems with robustness and speed trade-offs [3, 4, 5], optimal behavior can be predicted purely using analytical rigor. However, this is not always the case, and an empirical approach based on observed/calibrated data is the only recourse for predicting performance when analysis of the system is not feasible.

Recent years have also seen a steady adoption of component-based technologies for implementing complex scientific simulations [6, 7]. The idea of a monolithic parallel code is replaced by a collection of components, which may be composed into various feasible configurations. Further, these components may be adapted and/or replaced at runtime. Such an approach provides interoperability and flexibility, and can ideally be exploited to significantly improve application performance, especially in cases where the algorithmic and component behaviors as well as overall application execution are not known a priori. For example, a domain-decomposition component for parallel scientific simulations may adapt based on application state and particularly its heterogeneity. However, performing such runtime adaptation is non-trivial and requires an understanding of the requirements of a particular application state as well as a calibration of the impact of the adaptation on overall performance.

This paper uses runtime calibration to analyze the impact of computational heterogeneity on application performance, and investigates load balancing trade-offs when applied to different orchestrations of component-based scientific simulations. The analysis is based on a 2-D methane-air reaction-diffusion model solved using an operator-split method, which employs an iterative implicit time-integration scheme for the (stiff) reactive terms and creates a variable computational load. A domain-decomposition component for this simulation may be either based on the application structure such as domain geometry, or on the application characteristics such as pointwise varying workloads. The overarching goal of this research is to enable performance-enhancing runtime adaptations of the load balancing component based on the level of heterogeneity and application behavior. The characterization of computational heterogeneity and its performance implications, presented in this paper, are important steps towards achieving this objective.

 ISBN 1-56555-313-6

The rest of the paper is organized as follows. Section 2 provides an overview of scientific computing using components on structured grids. Section 3 outlines the component-based reaction-diffusion model that serves as the illustrative example in this paper. Section 4 describes the load balancing strategies deployed for partitioning high-performance combustion simulations. The runtime calibration mechanisms are also outlined in Section 4. Section 5 presents an experimental evaluation of the performance trade-offs for these load balancing strategies using uniform structured implementations for two different compositions of the combustion application. Section 6 presents concluding remarks.

2 SCIENTIFIC COMPUTING USING COMPONENTS ON STRUCTURED GRIDS

2.1 Parallel Scientific Simulations

Scientific simulations are modeled by systems of partial differential equations (PDEs) and are typically solved by uniform or adaptive discretization of the problem domain onto a structured or unstructured mesh/grid. The unknowns of the PDE are then approximated numerically at each discrete grid point. The resolution of the grid (or grid spacing) determines the local and global error of the approximation, and is typically dictated by the features of the solution that need to be resolved. The resolution of the grid also defines the computational costs and storage requirements of the simulation.

Parallel formulations of scientific applications on structured grids involve the decomposition of the problem domain among processors to balance load, either uniformly or based on application characteristics. Each processor then performs computations on its own grid blocks (locally owned regions of the application domain) and periodically synchronizes the data at block boundaries (maintained as "ghost" cells) with neighboring grid blocks. During synchronization, the ghost cells on each grid block are filled either via point-to-point processor communication (if the neighboring block is off-processor) or via local copying, both leading to complexity and overhead. It is generally hoped that these overheads stay low, but the ratio of the overheads to the computation is dependent on the problem size per processor, the nature and state of the network fabric, and the implementation details of the code being parallelized. The latter is often known during the parallelization/domain-decomposition effort.

Parallel implementations of scientific simulations are supported by several existing structured grid infrastructures that include Chombo [8], GrACE [9], and SAMRAI [10]. Note that though non-uniform load distributions occur in simulations on block-structured adaptive meshes, the load at each grid point is typically assumed to be homogeneous. However, non-uniform loads may also be a consequence of the numerical schemes used to conduct a simulation, even on a structured mesh, as is the case for pointwise varying computational loads presented in this paper.

2.2 Component-Based Scientific Computing

High-performance scientific simulations often leverage the combined expertise of multidisciplinary research teams comprising scientists, engineers, mathematicians, and computer scientists. However, code incompatibilities and lack of standardization among contributing teams lead to significant challenges in managing the complexity and performance of such simulation codes. As a result, component-based software engineering [11] has been widely adopted as the paradigm of choice in designing and managing large-scale scientific simulations. One such approach is the Common Component Architecture (CCA) [11] that offers greater flexibility, interoperability, and reuse in comparison to large, unwieldy monolithic codes.

The CCA paradigm envisages the creation of "components" [12], which generally embody a particular scientific model or a numerical, data-decomposition or I/O functionality. These components are designed with standard, clearly-defined interfaces that help to protect them from external changes in the software environment. Applications can be composed or assembled at runtime from components selected from a component pool. These selected components are peers and can interact with one another only through well-defined interfaces. Therefore, when an application needs to be modified, a single component can be modified (or exchanged for a similar component), without affecting the other components making up the application. Moreover, various compositions of the constituent components can result in different runtime scenarios for scientific applications.

3 THE REACTION-DIFFUSION SYSTEM

Gaseous, combusting flows, in the absence of significant sooting, consist of convection, diffusion and reactions involving a fuel, an oxidizer and a number of intermediate radicals. They are modeled by laws governing the conservation of mass, momentum, energy and the conservation of each of the radicals [13]. Transport processes refer to processes (typically, convection and diffusion) that transport various species (fuel, oxidizer or intermediate radicals) around in the domain of interest. The combustion model presented in this paper does not consider convection, and hence the transport processes are purely diffusive. On the other hand, reaction processes are processes that are governed by chemical reactions occuring in situ, i.e., based entirely on the state at a point. Reaction processes at two adjoining points interact with each other via transport processes.

In this study, we approximate the true behavior (for performance purposes) of this highly non-linear system by a set of reaction-diffusion partial differential equations (PDEs).

When solved with realistic chemical mechanisms, this PDE system is plagued by a wide spectrum of timescales, ranging from nanoseconds for reactive processes to tens of milliseconds for transport processes. A common technique to address this is operator-splitting [14] that advances chemistry implicitly, while transport may be dealt with either explicitly or implicitly. In either case, the integrators for chemistry and transport are very different. The implicit chemistry time-advancement leads to higher loads at grid points in reactive regions. A non-linear system is iteratively solved at each grid point; reactive regions with fast processes converge slowly as they have to be time-resolved. This leads to a spatially heterogeneous computational load. We consider the case of an igniting methane-air mixture and use the CFRFS [6, 15] Toolkit, which is a collection of CCA components, to simulate the problem. Two different chemical mechanisms are used, as described later in Section 5. Briefly, the reaction-diffusion system is of the form

$$\frac{\partial \phi_i}{\partial t} = \frac{\nabla P_i \cdot \nabla \phi_i}{Q_i} + R(\phi_j) \qquad (1)$$

$\phi_i, i = 1 \ldots N_{species} + 1$ at a grid point is the temperature or the mass fraction of $N_{species}$ chemical species at a given point in space. $R(\phi_j)$ models the production of heat and chemical species by reversible chemical reactions. Spatial derivatives are approximated using central finite differences. The case $i = 1$ corresponds to the temperature equation with $P_i = \rho C_p \alpha$ and $Q_i = \rho C_p$, where ρ is the mixture density, C_p is the specific heat at constant pressure of the mixture and α is the thermal diffusivity. For $i = 2 \ldots N_{species} + 1$, $P_i = \rho D_i$ and $Q_i = \rho$, where D_i is the diffusivity of each of the $N_{species}$ species. α and D_i are obtained from a mixture-averaged formulation. ρ, α, D_i and C_p couple the temperature and all the species together. Below, we will refer to $\phi_i, i = 1 \ldots N_{species} + 1$ as Φ. Equation 1 can be rewritten as

$$\frac{\partial \Phi}{\partial t} = T(\Phi) + R(\Phi) \qquad (2)$$

where $T(\Phi)$ contains all the spatial derivatives. The system is time-evolved in the following manner:

1. Over a timestep of $\Delta t_g/2$, we advance Φ^n (solution at timestep n) using $\Phi_t = T(\Phi)$ to Φ' with a second-order Runge-Kutta-Chebyshev scheme [6]. The gradients are computed using a central-difference scheme.

2. Using Φ' as initial condition, we solve $\Phi_t = R(\Phi)$ over Δt_g to Φ''. Since there are no spatial coupling terms, this system is solved on a point-by-point basis. At certain points, especially near flame fronts and ignition points, this ODE system exhibits very fast kinetics and has to be advanced using small timesteps (for accuracy reasons). This is done using BDF3 from the CVODE [16] package. This step accounts for the heterogeneity of workloads.

3. Using Φ'' as initial condition, we solve $\Phi_t = T(\Phi)$ over $\Delta t_g/2$ to get Φ^{n+1}. This is done exactly as in Step 1.

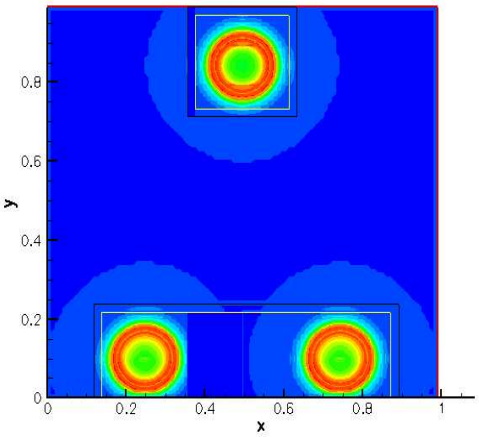

Figure 1. Illustrative snapshot of a 2-D methane-air combustion simulation with 3 hot-spots [17].

This scheme is also referred to as diffusion-reaction-diffusion (D-R-D) splitting. A R-D-R splitting is also possible and changes the computational load. The problem is solved in a square domain using a uniform (square) mesh. Zero-gradient boundary conditions (adiabatic system) are enforced. Three high-temperature Gaussian kernels are initialized in a stoichiometric methane-air mixture and serve as the ignition sites, as observed in Figure 1. Ignition fronts propagate out into the unburnt gas. Due to the wide spectrum of the timescales of the active (chemical) processes in the ignition front, the iterative solver embedded in the implicit time-integration scheme (Step 1 and 3 above) takes long to converge, as it resolves all the timescales. In contrast, the integration in the burnt and unburnt region is computationally light. Further, the computational load of the diffusion step (Step 1 above) is usually lighter than the reaction step. However, it involves ghost cell updates and incurs communication costs. Such an uneven loading of the domain requires a non-uniform decomposition across processors. Furthermore, the domain has to be redistributed as the ignition fronts propagate.

4 LOAD BALANCING FOR PARALLEL REACTION-DIFFUSION SIMULATIONS

Parallel simulations of the reaction-diffusion model exhibit conflicting load balancing requirements for reactive and transport processes. The degree of heterogeneity impacts the performance of the domain decomposition strategy and the

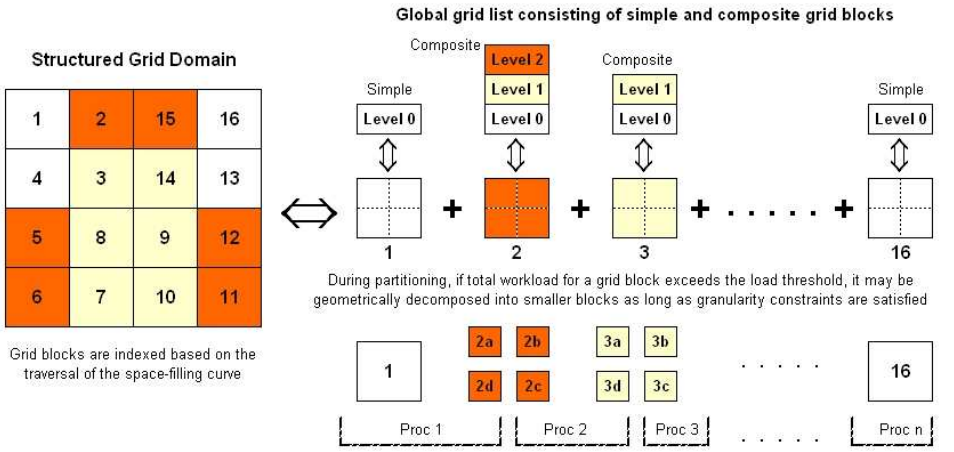

Figure 2. The structured grid domain is mapped to a global grid list for load balancing in Dispatch [17].

overall execution. In this section, we discuss load balancing schemes for structured meshes that address two extremes. *Blocked* distribution is a homogeneous scheme based on the domain geometry, whereas the *Dispatch* strategy addresses the computational heterogeneity. Note that these partitioning strategies are two different domain decomposition algorithms that have been integrated into the GrACE [9] infrastructure, which serves as the mesh and data management component in component-based implementations of high-performance scientific simulations.

4.1 *Blocked* Distribution

The *Blocked* [9] distribution strategy is based on application geometry and provides a spatially uniform decomposition along each axis of the structured domain. This widely-used scheme is relatively simple, has low overheads, and results in roughly the same number of grid points per processor when the domain can be favorably partitioned. This approach does not consider variations in computational heterogeneity and assumes equal load at each grid point. As a result, uniform resolution (unigrid) simulations do not perform domain redistribution for this scheme, since the domain structure remains unchanged.

4.2 *Dispatch* Strategy

Dispatch [17] combines inverse space-filling curve based partitioning (ISP) [9] with pointwise varying, in-situ global load balancing to address the dynamic partitioning and heterogeneous computational requirements of structured uniform or adaptive scientific applications.

The decomposition of the structured grid hierarchy is performed using space-filling curves (SFCs) [18]. SFCs are locality preserving recursive mappings from N-dimensional space to 1-dimensional space. *Dispatch* uses SFCs to map the entire application domain into a global grid list comprising grid blocks (representing sub-domains that keep all contained refinement levels), as illustrated in Figure 2. Moreover, *Dispatch* maintains the loads associated with pointwise reactive processes, which represent computational heterogeneity, using a workload grid function that is distributed among processors. Since *Dispatch* addresses computational heterogeneity that can change as the simulation progresses over time, this strategy is invoked at periodic intervals during application execution to perform redistribution and data remapping among processors.

During redistribution, the *Dispatch* scheme analyzes previous and current decompositions to determine the updated loads for existing grid blocks (representing portions of the application domain) and to generate interpolated workloads for new refinement regions in case of adaptive meshes. Each processor simultaneously computes and stores the loads for all grid blocks in its local work list and determines the cumulative workload for its local workload grid function. All processors construct a global work distribution list by concatenating individual local work lists and compute the global work threshold. The partitioning algorithm then decomposes the global grid list based on the threshold so that the total workload on each processor is nearly balanced. The load imbalance generated during this phase is due to granularity (minimum grid block dimension) and aspect ratio constraints that need to be satisfied. *Dispatch* balances the pointwise varying computational loads across processors and can result in complicated partitions with unequal number of grid points.

4.3 Runtime Calibration

To analyze the impact of computational heterogeneity on overall performance, we augment the application reaction component with sensors and timers that enable runtime cal-

ibration. The sensors profile the reaction characteristics and express heterogeneity in terms of the number of iterative solves performed at each pointwise process. The timers instrument the chemistry adapter (comprising the reaction compute section) at a fine level and measure the time spent in reaction time-advancement, implicit solves, and other data operations. During redistribution, the calibration component gathers the cumulative reaction computation times for all processors and computes the average, standard deviation, and coefficient of variation (defined as a dimensionless ratio of the standard deviation to the average) metrics for the heterogeneous loads. This coefficient of variation can serve as a useful indicator to analyze the impact of computational heterogeneity on runtime performance, especially in non-intuitive cases.

Note that the runtime calibration for component-based combustion simulations presented in this paper are performed by modifying the existing interfaces of various components. Though components should ideally be enhanced with specialized interfaces for obtaining application-specific performance characteristics, these are not available in current implementations, which require significant software engineering efforts to provide these calibration interfaces. For our proof-of-concept study, we focus on using the runtime calibration to analyze the impact of heterogeneity on load balancing and performance of parallel scientific simulations.

5 EXPERIMENTAL EVALUATION

The experimental evaluation is performed using structured unigrid implementations of the 2-D methane-air reaction-diffusion model. Two different orchestrations of this combustion application can be enabled by using either a reduced chemical mechanism (R-D kernel) or the more detailed/-accurate but computationally heavy GRI 1.2 [19] mechanism (CFRFS kernel). The variation in execution times for the two compositions can be attributed to the different computational requirements of the underlying chemical mechanisms.

Both models are evaluated on 64 processors of "Jacquard" [20] at the National Energy Research Scientific Computing Center (NERSC) using *Blocked* and *Dispatch* load balancing strategies, presented in Section 4. Jacquard is a 712-CPU Opteron cluster, arranged as 356 dual-processor nodes, running a Linux operating system. Each processor runs at a clock speed of 2.2GHz and has a theoretical peak performance of 4.4 GFlop/s. Processors on each node share 6GB of memory. The nodes are interconnected with a high-speed InfiniBand network.

The experiments consist of analyzing the computational heterogeneity and comparing the performance of the two partitioners by measuring overall application execution time, spans (defined as the difference between the maximum and minimum values) for reaction and diffusion compute time and synchronization time, and the redistribution overheads. Note that the synchronization time metric in this evaluation is the total time required to complete the synchronization operation and includes processor "wait" time as well as the time taken to perform ghost cell updates. Consequently, higher load imbalances among processors in the computation section can result in larger synchronization spans.

5.1 Methane-Air Model using a Reduced Chemical Mechanism

The experimental evaluation of the R-D kernel is performed on a 2-D uniform mesh with 512×512 resolution using a diffusion-reaction-diffusion (D-R-D) splitting strategy. A second-order central difference scheme is used to evaluate the spatial derivatives on the uniform mesh. A reduced methane-air mechanism with 25 species and 92 reversible reactions is used in this experiment. The application granularity is set to 4 and the simulation executes for 200 timesteps, with other application parameters kept unchanged. Due to splitting, a spatially variable load-distribution is achieved, as shown in Figure 3. The reactive processes near the flame fronts have high computational requirements that correspond to large values of workloads at the 3 hot-spots (ranging around 100-125), while in the bulk of the domain, they have a value near 1.

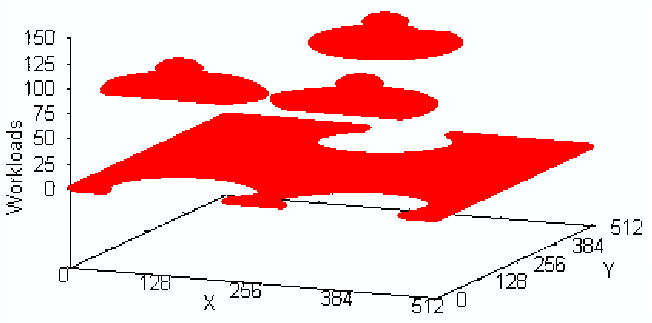

Figure 3. Computational heterogeneity at timestep 78 for the R-D simulation with 512*512 resolution.

Figure 4 illustrates the application execution times for the R-D kernel obtained with *Blocked* and *Dispatch* load balancing strategies. The average redistribution time as well as the spans for reaction and diffusion compute times and application synchronization (sync) times are also shown in Figure 4. Table 1 presents further details on the performance metrics and the heterogeneity analysis for the R-D simulation, viz. the average, standard deviation and coefficient of variation for reaction computation times (μ^R_{comp}, σ^R_{comp}, CV^R_{comp}), diffusion computation times (μ^D_{comp}, σ^D_{comp}, CV^D_{comp}), and synchronization times (μ_{sync}, σ_{sync}, CV_{sync}). Even though the R-D kernel uses a reduced chemical mechanism, the simulation exhibits large variation in computational heterogeneity, as observed in Figure 3, and deduced from the high values of reaction

span and CV^R_{comp}. The calibration component reports that the pointwise reactive loads vary by an order of nearly 100-125, and the coefficient of variation for the reaction component exceeds 100%.

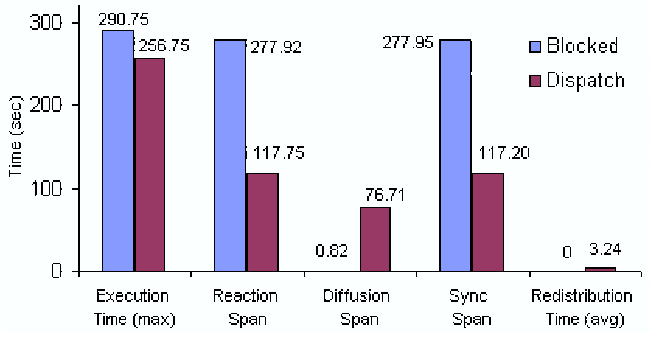

Figure 4. Performance evaluation of *Blocked* and *Dispatch* load balancing strategies for R-D kernel on 64 processors.

In this scenario, the *Blocked* scheme performs decomposition based on domain geometry that results in very high imbalance in the reaction component. This high imbalance is also reflected in the large sync span, since lightly-loaded processors have to wait during the synchronization stage for the overloaded processors to complete their local computations. However, the diffusion component is well-balanced by the *Blocked* strategy since the diffusion phenomenon is based on the number of grid points rather than the load at each point. Also, there are no redistribution costs since the domain structure remains unchanged.

The R-D performance of *Dispatch* is antipodal to that of the *Blocked* strategy. *Dispatch* considers the variation in computational heterogeneity and improves overall R-D performance by about 12%, while maintaining low redistribution overheads (1.26% of total execution time). While *Dispatch* provides better load balance and improved sync times (evident from the low values of CV^R_{comp} and CV_{sync}), the diffusion compute times suffer due to the non-equitable distribution of grid points. However, the magnitude and variation in heterogeneity within the reaction component overshadow the runtime effects of diffusion, and *Dispatch* provides overall better performance as compared to the *Blocked* scheme. Therefore, the *Dispatch* partitioner is well-suited for load balancing parallel simulations of the R-D methane-air model.

5.2 Methane-Air Model using GRI 1.2

This experiment is conducted on a 2-D uniform mesh with 500×500 resolution using the CFRFS [6] Toolkit, a component-based toolkit for simulating reacting flows. The simulation executes for 20 timesteps, with other application parameters kept unchanged. We use a reaction-diffusion-reaction (R-D-R) splitting, unlike Section 5.1, which increases the overall computational load due to the two reaction steps. The spatial derivatives in the diffusion step are computed using fourth-order central differences - this also requires one to keep a broader border of ghost cells on each processor and, hence, the application granularity is set to 8. The diffusion coefficients are computed using DRFM [21]. The GRI 1.2 chemical mechanism [19] is used in this evaluation and consists of 32 species and 177 reversible reactions.

Table 1. Heterogeneity analysis for *Blocked* and *Dispatch* schemes applied to different models of the 2-D methane-air combustion application on 64 processors.

Runtime Parameters	R-D kernel		CFRFS kernel	
	Blocked	*Dispatch*	*Blocked*	*Dispatch*
μ^R_{comp} (s)	77.13	81.6	1423.18	1590.02
σ^R_{comp} (s)	102.14	28.48	108.19	200.3
CV^R_{comp}	1.327	0.349	0.076	0.126
μ^D_{comp} (s)	9.03	9.15	279.42	314.73
σ^D_{comp} (s)	0.14	15.64	10.01	40.29
CV^D_{comp}	0.016	1.709	0.036	0.128
μ_{sync} (s)	204.42	161.69	13.57	95.94
σ_{sync} (s)	102.11	27.77	9.73	31.09
CV_{sync}	0.5	0.172	0.717	0.324

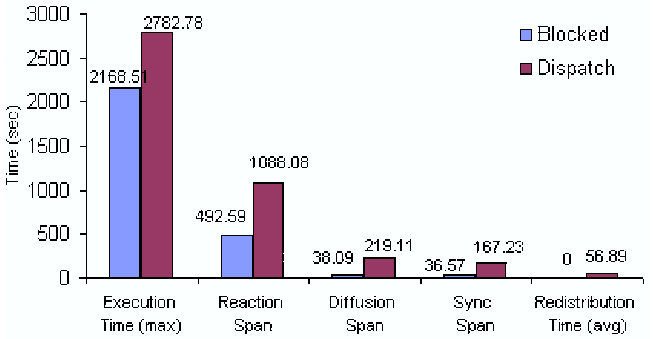

Figure 5. Performance evaluation of *Blocked* and *Dispatch* load balancing schemes for CFRFS kernel on 64 processors.

The computational load due to chemistry is estimated to be nearly five times more than due to diffusion. Intuitively, one may posit that *Dispatch* should outperform the *Blocked* scheme, based on the empirical analysis of the R-D kernel presented in Section 5.1. However, this is not the case, as illustrated in Figure 5 that shows the comparative performance of the two load balancing strategies for the CFRFS kernel. The *Blocked* partitioner improves overall application performance by nearly 22% as compared to *Dispatch*. *Dispatch* exhibits larger spans for reaction and diffusion compute times and synchronization times, but maintains low redistribution overheads (2% of the total execution time).

Table 1 presents the runtime performance metrics and the

heterogeneity analysis for the CFRFS simulation. Though the reaction component dominates diffusion in terms of computation time (μ^R_{comp} is roughly five times greater than μ^D_{comp}), there is low variation among the pointwise varying loads apparent from the low values of CV^R_{comp} (7.6% for *Blocked* and 12.6% for *Dispatch*). Similar low values are observed for CV^D_{comp} as well. The calibration component reports that the pointwise loads differ by a factor of 2 approximately, which is very low compared to the large variation observed in the R-D kernel. Moreover, the reaction coefficient of variation computed by the calibration component ranges around 4-7% during the course of the simulation. In such simulations with relatively low heterogeneity, *Dispatch* may well prove to be an overkill due to more complicated partitions and possibly higher imbalance that can increase both reaction and diffusion component times, resulting in degraded performance. This indicates that the evaluated CFRFS kernel may be better suited to partitioning schemes based on domain geometry, since the simulation has relatively homogeneous computational characteristics, even though the reaction component is quite compute-intensive.

5.3 Inferences

In the experimental evaluation presented above, R-D and CFRFS are combustion kernels that exhibit different execution times due to different complexities of the chemical mechanisms. Moreover, runtime heterogeneity impacts the performance of the domain decomposition component for both kernels in different ways. It can, therefore, be inferred that the overall performance of component-based scientific simulations is highly dependent on the problem characteristics and the implementation and particular connectivity of the components in the simulation code, none of which are known before runtime. This study illustrates these uncertainties that can be introduced by componentization as well as heterogeneity, and motivates the need for an empirical approach using runtime calibration, heuristics or prediction to ensure a degree of performance in such poorly characterized environments.

The research presented in this paper motivates the investigation of a hybrid domain decomposition approach that can select an appropriate load balancing component (from a pool of available load balancers) at runtime, based on calibration of heterogeneity and performance for a particular orchestration of the component-based simulation. Such an approach can be augmented with feedback and other control-theoretic methods to perform runtime adaptations, thereby reducing the uncertainties due to component behaviors and improving overall simulation performance.

6 CONCLUSION

This paper analyzed the impact of computational heterogeneity on application performance, and investigated partitioning and load balancing trade-offs. Specifically, it presented a mechanism for determining the computational heterogeneity at runtime and calibrated the performance of *Blocked* and *Dispatch* load balancing strategies. This research is motivated by the requirements for exploiting the flexibility offered by component-based technologies, which include the ability to calibrate component behavior and performance at runtime. Experimental evaluations using 2-D methane-air combustion simulations demonstrate that an empirical analysis of computational heterogeneity can provide useful insights, which can be used to improve performance, especially in cases where the behaviors and overall execution are not known a priori. Future research aims at investigating runtime characterization and adaptation for parallel scientific components using a robust control system with policies and feedback to improve application performance.

ACKNOWLEDGMENTS

We thank Dr. John Hewson, Sandia National Laboratories, Albuquerque, NM, for his help with the methane-air chemical mechanism and generating a self-supporting chemistry module for us. The research presented in this paper is supported in part by National Science Foundation via grants numbers ACI 9984357, EIA 0103674, EIA 0120934, ANI 0335244, CNS 0305495, CNS 0426354 and IIS 0430826, and by the US Department of Energy via the grant number DE-FG02-06ER54857. This work is also supported by the US Department of Energy (DOE), Office of Science's SciDAC program. Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

REFERENCES

[1] Chandra, S., S. Sinha, M. Parashar, Y. Zhang, J. Yang, and S. Hariri. 2002. "Adaptive Runtime Management of SAMR Applications." In *Proceedings of the 9th International Conference on High Performance Computing*, Bangalore, India, Springer Verlag, vol. 2552, 564-574.

[2] Berger, M. J. and P. Collela. 1989. "Local Adaptive Mesh Refinement for Shock Hydrodynamics." *Journal of Computational Physics*, vol. 82, 64-84.

[3] Bhowmick, S., P. Raghavan, L. C. McInnes, and B. Norris. 2004. "Faster PDE-Based Simulations using Robust Composite Linear Solvers." *Future Generation Computer Systems*, vol. 20, 373-387.

[4] Bhowmick, S., L. C. McInnes, B. Norris, and P. Raghavan. 2003. "The Role of Multi-Method Linear Solvers in PDE-based Simulations." *Computational Science and its Applications - ICCSA 2003*, Lecture Notes in Computer Science, vol. 2667, 828-839.

[5] McInnes, L. C., B. Norris, S. Bhowmick, and P. Raghavan. 2003. "Adaptive Sparse Linear Solvers for Implicit CFD using Newton-Krylov Algorithms." In *Proceedings of the 2nd MIT Conference on Computational Fluid and Solid Mechanics*, Cambridge, MA, Elsevier, vol. 2, 1024-1028.

[6] Lefantzi, S., J. Ray, C. A. Kennedy, and H. N. Najm. 2005. "A Component-based Toolkit for Reacting Flows with High Order Spatial Discretizations on Structured Adaptively Refined Meshes." *Progress in Computational Fluid Dynamics*, vol. 5, no. 6, 298-315.

[7] Kenny, J. P., S. B. Benson, et al. 2004. "Component-based Integration of Chemistry and Optimization Software." *Journal of Computational Chemistry*, vol. 25, 1717-1725.

[8] Colella, P.; et al. 2005. Chombo Infrastructure for Adaptive Mesh Refinement. http://seesar.lbl.gov/ANAG/-chombo/.

[9] Parashar, M., J. C. Browne, C. Edwards, and K. Klimkowski. 1997. "A Common Data Management Infrastructure for Adaptive Algorithms for PDE Solutions." In *Proceedings of the Supercomputing Conference*, San Jose, CA, 1-22.

[10] Kohn, S. 2005. SAMRAI: Structured Adaptive Mesh Refinement Applications Infrastructure. http://-www.llnl.gov/CASC/SAMRAI/.

[11] Bernholdt, D. E., B. A. Allan, R. Armstrong, et al. 2006. "A Component Architecture for High-Performance Scientific Computing." *International Journal of High-Performance Computing Applications*, vol. 20, 162-202.

[12] CCA Forum. 2004. Common Component Architecture Forum. http://www.cca-forum.org.

[13] Najm, H. N., R. W. Schefer, R. B. Milne, C. J. Mueller, K. D. Devine, and S. N. Kempka. 1998. "Numerical and Experimental Investigation of Vortical Flow-Flame Interaction." Sandia Technical Report SAND98-8232, UC-1409, Sandia National Laboratories, Livermore, CA.

[14] Knio, O. M., H. N. Najm, and P. S. Wyckoff. 1999. "A Semi-Implicit Numerical Scheme for Reacting Flow II - Stiff, Operator-Split Formulation." *Journal of Computational Physics*, vol. 154, 428-467.

[15] Lefantzi, S., J. Ray, and H. N. Najm. 2003. "Using the Common Component Architecture to Design High Performance Scientific Simulation Codes." In *Proceedings of the International Parallel and Distributed Processing Symposium*, Nice, France, IEEE Computer Society Press.

[16] Cohen, S. D. and A. C. Hindmarch. 1996. "CVODE, a stiff/nonstiff ODE solver in C." *Computers in Physics*, vol. 10, no. 2, 138-143.

[17] Chandra, S., M. Parashar, and J. Ray. 2006. "Dynamic Structured Partitioning for Parallel Scientific Applications with Pointwise Varying Workloads." In *Proceedings of the 20th International Parallel and Distributed Processing Symposium*, Rhodes Island, Greece, IEEE Computer Society Press.

[18] Sagan, H. 1994. *Space filling curves*, Springer-Verlag.

[19] Frenklach, M., H. Wang, et al. 1995. "GRI-Mech— An Optimized Detailed Chemical Reaction Mechanism for Methane Combustion." Technical Report GRI-95/0058, Gas Research Institute, Chicago, IL.

[20] NERSC. 2006. Jacquard Home Page. http://-www.nersc.gov/nusers/resources/jacquard.

[21] Paul, P. H. 1997. "DRFM: A New Package for the Evaluation of Gas-Phase-Transport Properties." Sandia Technical Report SAND98-8203, Sandia National Laboratories, Albuquerque, NM.

ABOUT THE AUTHORS

Sumir Chandra is a Ph.D. student in the Department of Electrical and Computer Engineering and a researcher at the Center for Advanced Information Processing at Rutgers University in Piscataway, NJ. He has been a visiting researcher at Sandia National Laboratories in Livermore, CA. He received a B.E. degree in Electronics Engineering from Bombay University, India and a M.S. degree in Computer Engineering from Rutgers University. His research interests include high performance scientific computing, modeling and simulation, software engineering, and performance optimization.

Manish Parashar is Professor of Electrical and Computer Engineering at Rutgers University, where he also is co-director of the Center for Advanced Information Processing and director of the Applied Software Systems Laboratory. He received a B.E. degree in Electronics and Telecommunications from Bombay University, India and M.S. and Ph.D. degrees in Computer Engineering from Syracuse University. He has received the Rutgers Board of Trustees Award for Excellence in Research (2004-2005), NSF CAREER Award (1999) and the Enrico Fermi Scholarship from Argonne National Laboratory (1996). His research interests include autonomic computing, parallel and distributed computing, scientific computing, and software engineering.

Jaideep Ray is a Principal Member of Technical Staff in the Advanced Software Research and Development group at Sandia National Laboratories in Livermore, CA. He received a B.S. degree in Aerospace Engineering from Indian Institute of Technology, Kharagpur, India and M.S. and Ph.D. degrees in Mechanical and Aerospace Engineering from Rutgers, The State University of New Jersey. His research interests include fluid dynamics, numerical methods and scientific computing.