Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE - FUNDAMENTAL

Challenges and Opportunities for the Simulation of Calcium Waves on Modern Multi-Core and Many-Core Parallel Computing Platforms

Carlos Barajas | Matthias K. Gobbert | Gerson C. Kroiz | Bradford E. Peercy

[1]Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD 21250, USA

Correspondence
Matthias K. Gobbert, Department of Mathematics and Statistics, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA. Email: gobbert@umbc.edu

Abstract

State-of-the-art distributed-memory computer clusters contain multi-core CPUs with 16 and more cores. The second-generation of the Intel Xeon Phi many-core processor has more than 60 cores with 16 GB of high-performance on-chip memory. We contrast the performance of the second-generation Intel Xeon Phi, code-named Knights Landing (KNL), with 68 computational cores to the latest multi-core CPU Intel Skylake with 18 cores. A special-purpose code solving a system of nonlinear reaction-diffusion partial differential equations with several thousands of point sources modeled mathematically by Dirac delta distributions serves as realistic test bed. The system is discretized in space by the finite volume method and advanced by fully implicit time-stepping, with a matrix-free implementation that allows the complex model to have an extremely small memory footprint. The sample application is a seven variable model of calcium induced calcium release (CICR) that models the interplay between electrical excitation, calcium signaling, and mechanical contraction in a heart cell. The results demonstrate that excellent parallel scalability is possible on both hardware platforms, but that modern multi-core CPUs outperform the specialized many-core Intel Xeon Phi KNL architecture for a large class of problems such as systems of parabolic partial differential equations.

KEYWORDS:
Intel Xeon Phi, Intel Skylake, calcium induced calcium release, reaction-diffusion equations, finite volume method, strong scalability

1 | INTRODUCTION

This paper is concerned with demonstrating the scalability that is possible with parallel computer code both on modern multi-core CPUs and many-core processors such as the Intel Xeon Phi. The demonstration uses an appropriate and modern numerical method for a system of coupled, non-linear, time-dependent partial differential equations (PDEs) that has the form of reaction-diffusion equations

$$\frac{\partial u^{(i)}}{\partial t} = \nabla \cdot \left(D^{(i)} \nabla u^{(i)} \right) + q^{(i)}(u^{(1)}, \dots, u^{(n_s)}, \mathbf{x}, t) \quad i = 1, \dots, n_s, \tag{1}$$

for functions $u^{(i)} = u^{(i)}(\mathbf{x}, t)$, $i = 1, \dots, n_s$, of space $\mathbf{x} \in \Omega \subset \mathbb{R}^3$ and time $0 \leq t \leq t_{fin}$ of the n_s variables. The diffusivity matrices $D^{(i)} = \text{diag}(D^{(i)}_{11}, D^{(i)}_{22}, D^{(i)}_{33}) \in \mathbb{R}^{3\times 3}$ consist of positive entries. The forcing terms $q^{(i)}$ in (1) contain the non-linearities in the application problem of calcium waves in a heart cell as well as many thousands of point sources, modeled mathematically by Dirac delta distributions, on a large lattice of locations throughout the cell, at which calcium ions can be injected; this injection is turned on and off repeatedly throughout the simulations, based on a probabilistic model. Despite the fundamentally smooth behavior of this reaction-diffusion model, these crucial features of the application are responsible for many of the challenges: (i) The numerical method used as spatial discretization will not be convergent to as high an order as conventional, since the source functions are not sufficiently smooth [1,2]. (ii) The point sources are the crucial driver of the physiological effects, thus the domain of the cell needs to be discretized with a fine mesh to accommodate the thousands of point sources [3,4]. (iii) Sophisticated time-stepping is needed to resolve the switching mechanism over time [3,4]. (iv) The code needs to be parallelized to reach desired large final times t_{fin}, ideally eventually on the scale of laboratory experiments of several minutes. — It is point (iv) that we focus here in this paper since equations of this type occur often in mathematical biology and are thus a good example to demonstrate the power of parallel computing. Specifically, the equations are of parabolic type with key properties of mass conservation and smoothing behavior over time. The finite volume method (FVM) is the most appropriate discretization for this problem, since it guarantees mass conservation at the discrete level and will allows the incorporation of an advection term [2]; by contrast, the finite element conserves mass, but needs more complicated handling of advection.

The concrete test code used in the demonstration is a special-purpose code that is programmed in C with MPI as parallel library for one model of calcium waves in a heart cell. To make such simulations feasible, in particular for higher mesh resolutions and for longer simulation times, an efficient parallellization of the code as well as state-of-the-art computer hardware are necessary. This work analyzes the performance of our special-purpose code for this application on two currently available hardwares, namely of latest multi-core Intel Skylake CPUs with 18 cores and the second-generation many-core Intel Xeon Phi, code-named Knights Landing (KNL), with 68 computational cores. The simulations bear out that long-time studies for modest meshes are possible up to moderate times like 1,000 ms. Speedup for larger meshes is very good up to large numbers of nodes, and using many can facilitate simulations for fine meshes. This situation poses opportunities for more research into the numerical algorithms, their implementation, as well as the choice of computer hardware. Our results also bring out that the second-generation Intel Xeon Phi Knights Landing (KNL), despite its promise with large on-chip memory, is not competitive with more customary CPU-based nodes, though.

This paper is organized as follows: Section 2 details the application problem and the seven variable model used in this work. Section 3 specifies the numerical method used including its parallelization and implementation. Section 4 contrasts the multi-core Skylake and the and many-core KNL processors. Section 5 presents both sample application results as well as the parallel performance studies. on the multi-core and many-core processors. Section 6 collects the conclusions and discusses future work.

2 | DYNAMICS OF A CARDIAC CELL

The leading cause of death in the United States is currently heart disease[5]. In order to continue searching for methods to combat heart disease, it is vital that the heart and its underlying processes are understood with greater depth. The importance of having a greater understanding of the heart provides the motivation for this research.

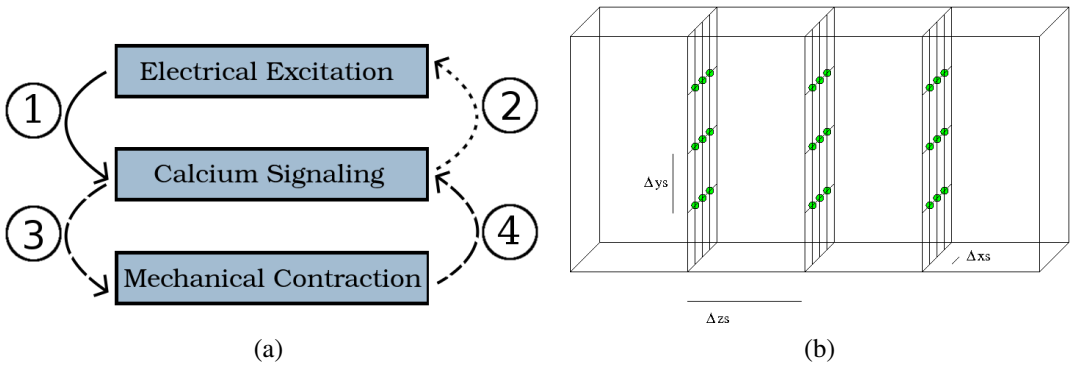
The line of work of this project focuses on a single cardiac cell and uses a mathematical model in order to represent the electrical excitation, calcium signaling, and mechanical contraction components of a cardiomyocyte. The original model for calcium induced calcium release (CICR) was introduced by Izu and co-workers[6,7] with a three variable model and included only calcium signaling. This original model comprises the heart of the Calcium Signaling component of the system indicated in Figure 1 (a). Figure 1 (b) sketches the domain of the simulation. A cell is reasonably modeled by the hexahedral shape elongated in the z-direction, since the focus of CICR research is on estimating reasonable physiological parameter values. A key of the model[6,7] is to place the calcium release units (CRUs) on a regular lattice of size $15 \times 15 \times 31 = 6{,}975$ throughout the cell; Figure 1 (b) shows three z-planes of 3×3 CRUs as example. At these many locations, calcium ions are released from the calcium store in the sarcoplasmic reticulum (SR) into the cytosol of the cell, and CRUs can open ('fire') and close repeatedly over time[6,7].

The model was extended for the first time to include the Electrical Excitation component in Figure 1 (a) by Alexander et al.[8], which implemented a one-way interaction from electrical excitation to calcium signaling indicated by link ① in Figure 1 (a). Studies with six variables by Angeloff et al.[9] extended the coupling to include a two-way cycle between electrical excitation and calcium signaling by incorporating both links ① and ② in Figure 1 (a). The 2016 paper by Angeloff et al.[9] also introduced the formulation of the complete eight variable model for all components in Figure 1 (a), but not all model variables were used in the simulations, and the studies did not incorporate the mechanical system.

Work by Deetz et al.[10,11] contained the first simulations that include the Mechanical Contraction component in Figure 1 (a) by activating the links ③ and ④ in Figure 1 (a). This is facilitated by adding a buffer species whose concentration can be related to the contraction of the cell. Thus, this work uses a model with seven variables to represent the excitation-contraction coupling (ECC) occurring in the cardiomyocyte, in which CICR is the mechanism through which electrical excitation is coupled with mechanical contraction through calcium signaling. The results of the simulations by Deetz et al.[10,11] allow us to draw two key conclusions about the extension of the model and its implementation: The model is capable of connecting the voltage to the contraction of the heart cell via the cytosol calcium and the third buffer species; and a stronger coupling strength from voltage to calcium leads to stronger contraction.

More recent studies by Kroiz et al.[12] investigate the successive introduction of each of the components in Figure 1 (a) step-by-step using an improved version of the seven variable model. This sets the stage for even longer-term simulation up to larger final times; these simuations crucially take advantage of the efficiency studies presented in this paper for the selection of parallel computing environments and their settings, thus showing how crucial efficient code is to further the modeling work.

The remainder of this section provides more details of the concrete seven variable model of the form (1) used in this work. The seven variables of the model are calcium in the cytosol $c(\mathbf{x}, t)$, a free or unbound florescent dye $b_1^{(c)}(\mathbf{x}, t)$, a free or unbound contractile protein (troponin) $b_2^{(c)}(\mathbf{x}, t)$, the inactive actin-myosin cross-bridges $b_3^{(c)}(\mathbf{x}, t)$, calcium in the sarcoplasmic reticulum (SR) $s(\mathbf{x}, t)$, voltage $V(\mathbf{x}, t)$, and the potassium gating function $n(\mathbf{x}, t)$. The variables with their units are listed in Table 1 and are



(a) (b)

FIGURE 1 (a) The three components of the model and their links labeled ① to ④. (b) The CRU lattice with spacings Δx_s, Δy_s, Δz_s throughout the three-dimensional domain.

modeled by the system of seven time-dependent, coupled, non-linear reaction diffusion equations

$$\frac{\partial c}{\partial t} = \nabla \cdot (D_c \nabla c) + R_1^{(c)} + R_2^{(c)} + \left(J_{CRU} + J_{leak} - J_{pump}\right) + \kappa\, J_{LCC} + \left(J_{m_{leak}} - J_{m_{pump}}\right), \tag{2}$$

$$\frac{\partial b_1^{(c)}}{\partial t} = \nabla \cdot (D_{b_1^{(c)}} \nabla b_1^{(c)}) + R_1^{(c)}, \tag{3}$$

$$\frac{\partial b_2^{(c)}}{\partial t} = \nabla \cdot (D_{b_2^{(c)}} \nabla b_2^{(c)}) + R_2^{(c)}, \tag{4}$$

$$\frac{\partial b_3^{(c)}}{\partial t} = \nabla \cdot (D_{b_3^{(c)}} \nabla b_3^{(c)}) + R_3^{(c)}, \tag{5}$$

$$\frac{\partial s}{\partial t} = \nabla \cdot (D_s \nabla s) - \gamma (J_{CRU} + J_{leak} - J_{pump}), \tag{6}$$

$$\frac{\partial V}{\partial t} = \nabla \cdot (D_v \nabla V) + \tau_v \frac{1}{C}\left[I_{\text{app}} - g_L(V - V_L) - g_{Ca}\, m_\infty(V)\,(V - V_{Ca}) - g_K\, n\,(V - V_K) - \omega\,(J_{m_{leak}} - J_{m_{pump}})\right], \tag{7}$$

$$\frac{\partial n}{\partial t} = \nabla \cdot (D_n \nabla n) + \tau_v\, \lambda_n \cosh\left(\frac{V - V_3}{2V_4}\right)\left(n_\infty(V) - n\right) \tag{8}$$

The following provides details on each of the components of the model, the calcium signaling portion of the model, the electrical excitation that is connected to the calcium signaling in both the feedforward and feedback directions represented by link ① and link ② in Figure 1 (a), and the mechanical contraction component that is also connected to the calcium signaling in both the feedback and feedforward directions represented by links ③ and ④ in Figure 1 (a).

Table 1 collects the variables of the model with their units as well as their initial values. The coefficients D_c, $D_{b_i^{(c)}}$, D_s, D_v, and D_n are the diffusivity matrices for each variable [12]; here, D_v is modeled as a high value for near-instantaneous distribution of voltage through the cell, and $D_n = 0$ which localizes the effect of the potassium channels.

TABLE 1 Independent and dependent variables of the seven variable model and their initial conditions. The concentration unit M is shorthand for mol/L (moles per liter).

Variable	Definition	Initial value and units
$\mathbf{x}$	spatial position variable (x, y, z)	μm
t	time variable	ms
$c(\mathbf{x}, t)$	calcium in the cytosol	$c_0 = 0.1$ μM
$b_1^{(c)}(\mathbf{x}, t)$	free flourescent dye in the cytosol	45.918 μM
$b_2^{(c)}(\mathbf{x}, t)$	free troponin in the cytosol	111.818 μM
$b_3^{(c)}(\mathbf{x}, t)$	inactive actin-myosin cross-bridges [X]	145.20 μM
$s(\mathbf{x}, t)$	calcium in the SR	$s_0 = 10{,}000$ μM
$V(\mathbf{x}, t)$	membrane potential (voltage)	-50 mV
$n(\mathbf{x}, t)$	fraction of open potassium channels	0.1

The calcium signaling portion of the model consists of the equations (2)–(6). The reaction terms $R_i^{(c)}$ describe the reactions between calcium and the buffer species. They are the connections between (2)–(5). More precisely, we have

$$R_1^{(c)} = -k^+_{b_1^{(c)}}\, c\, b_1^{(c)} + k^-_{b_1^{(c)}}\left(b^{(c)}_{1,total} - b_1^{(c)}\right), \tag{9}$$

for reaction between cytosol calcium and the flourescent dye.

When troponin binds to Ca^{2+}, the protein as a whole changes shape: this not only allow actin-myosin cross-bridges to form, but also traps the calcium in its connection to the troponin so that the disassociation rate decreases dramatically. To account for this, the shortening factor ε describes how the separation of troponin and calcium has been physically, but not chemically, impaired. Note, again, that $R_2^{(c)}$ remains a function of cytosol calcium concentration $c(\mathbf{x}, t)$ by its equation

$$R_2^{(c)} = -k^+_{b_2^{(c)}}\, c\, b_2^{(c)} + k^-_{b_2^{(c)}}\left(b^{(c)}_{2,total} - b_2^{(c)}\right)\frac{1}{\varepsilon} \tag{10}$$

with

$$\varepsilon = \exp\left(F_{max}\, k_s \left(\frac{b^{(c)}_{3,total} - b^{(c)}_3 - [XB]_0}{b^{(c)}_{3,total} - [XB]_0}\right)\right) \tag{11}$$

and $[XB]_0 = b^{(c)}_{3,total} - b^{(c)}_3(\mathbf{x}, 0)$. This shortening factor ε links ③ and ④ in Figure 1 (a). It refers back to the concentration of $b^{(c)}_3(\mathbf{x}, t)$, the inactive actin-myosin cross-bridges, and to the force that their linkage generates. It is scaled by the maximum possible contractile force F_{max}, the actin stiffness k_s, and the proportion of active to inactive actin-myosin cross-bridges.

The contractile proteins in question, though considered as a single species, are the combination of actin and myosin when linked via cross-bridges. This linkage is made possible by Ca^{2+} binding to troponin, the cytosol buffer species $b^{(c)}_2(\mathbf{x}, t)$: it is this binding that allows the actin-myosin cross-bridges to form. The cytosol species, $b^{(c)}_3(\mathbf{x}, t)$, describes these actin-myosin cross-bridges and constructs a third cytosol reaction term

$$R^{(c)}_3 = -k^+_{b^{(c)}_3}\left(\frac{b^{(c)}_{2,total} - b^{(c)}_2}{b^{(c)}_{2,total}}\right)^2 b^{(c)}_3 + k^-_{b^{(c)}_3}(b^{(c)}_{3,total} - b^{(c)}_3). \tag{12}$$

Notice that this is not the same as the generic pattern for buffer species reaction terms from the initial model. There is no immediately clear dependence on cytosolic calcium $c(\mathbf{x}, t)$. However, while $c(\mathbf{x}, t)$ is not explicitly included, it is present in the proportion involving troponin, $b^{(c)}_3(\mathbf{x}, t)$, which itself depends explicitly on cytosol calcium levels; $R^{(c)}_3$, like the other two reaction equations, does in fact depend on the cytosol calcium concentration.

These two reaction terms (10) and (12) connect the three components of our model. The calcium signaling is linked to the pseudo-mechanical contraction through the cross-bridge term, and the pseudo-mechanical contraction is in turn connected to the calcium signaling through the inclusion of the cytosol calcium concentration in the modified reaction equation for troponin. Thus all links ①, ②, ③, and ④ in Figure 1 (a) are established, and thus the three systems of the model are fully linked.

Note that in the reaction rates (9) and (10), ${b_i}^{(c)}$ is the amount of unbound buffer known as "free" buffer. The constant $b^{(c)}_{i,total}$ denotes the total bound and unbound calcium thus leaving the difference seen in (9) and (10) to be the bound calcium. Since the model uses no-flux boundary conditions, no buffer species escapes or enters the cell, thus we only need to track the "free" buffer species and use $b^{(c)}_{i,total} - {b_i}^{(c)}$ for the bound species.

The flux terms J_{CRU}, J_{leak}, and J_{pump} in (2) describe the calcium induced release of Ca^{2+} into the cytosol from the SR, the continuous leak of Ca^{2+} into the cytosol from the SR, and the pumping of Ca^{2+} back into the SR from the cytosol. The terms J_{LCC}, $J_{m_{leak}}$, and $J_{m_{pump}}$ describe the fluxes of calcium into and out of the cell via the plasma membrane. The coupling between (2) and (6) is achieved by the three flux terms shared by both.

More precisely, J_{LCC}, $J_{m_{leak}}$, and $J_{m_{pump}}$ in (2) describe the fluxes of calcium into and out of the cell via the plasma membrane. J_{pump} replenishes the calcium stores in the SR; it increases SR calcium concentration by decreasing cytosol calcium concentration. J_{leak} is a continuous leakage of those SR calcium stores into the cytosol; it increases cytosol concentration by decreasing SR calcium concentration. The pump term

$$J_{pump}(c) = V_{pump}\left(\frac{c^{n_{pump}}}{K^{n_{pump}}_{pump} + c^{n_{pump}}}\right) \tag{13}$$

is thus a function of cytosol calcium $c(\mathbf{x}, t)$. and consists of the maximum pump velocity V_{pump} multiplied against the relationship between $c(\mathbf{x}, t)$ and the pump sensitivity K_{pump}; the exponent n_{pump} refers to the Hill coefficient (quantifying the degree of cooperative binding) for the pump function. This has the practical effect of multiplying the maximum possible pump velocity against a number between 0 and 1, exclusive. The leak term J_{leak} is a constant defined by

$$J_{leak} = J_{pump}(c_0), \tag{14}$$

which balances $J_{pump}(c)$ at basal level $c_0 = 0.1\ \mu$M of cytosol calcium. As noted, J_{pump} has two roles, namely to balance J_{leak} in the absence of sparking, but also to balance J_{CRU} under conditions of active calcium release.

The term J_{CRU} in (2) is the Ca^{2+} flux into the cytosol from the SR via each individual point source at which a CRU has been assigned. The effect of all CRUs is modeled as a superposition such that

$$J_{CRU}(c, s, \mathbf{x}, t) = \sum_{\hat{\mathbf{x}} \in \Omega_s} \hat{\sigma}\, \frac{s(\mathbf{x}, t)}{s_0}\, \mathcal{O}(c, s)\, \delta(\mathbf{x} - \hat{\mathbf{x}}) \tag{15}$$

with

$$\mathcal{O}(c, s) = \begin{cases} 1 & \text{if } u_{rand} \leq J_{prob}, \\ 0 & \text{if } u_{rand} > J_{prob}, \end{cases} \tag{16}$$

where

$$J_{prob}(c, s) = P_{max} \left(\frac{c^{n_{prob_c}}}{K_{prob_c}^{n_{prob_c}} + c^{n_{prob_c}}} \right) \left(\frac{s^{n_{prob_s}}}{K_{prob_s}^{n_{prob_s}} + s^{n_{prob_s}}} \right). \tag{17}$$

The effect of each CRU is modeled here as a product of three terms: (i) Similarly to how in J_{pump} the maximum pump rate is scaled against the concentration of available cytosol calcium, the maximum pump rate is scaled against the concentration of available cytosol calicum, the maximum rate of Ca^{2+} release $\hat{\sigma}$ is scaled here against the ratio of calcium concentration in the SR. (ii) Following the same pattern a maximum value multiplied against some scaling proportion between 0 and 1 the gating function $\mathcal{O}(c, s)$ has the practical effect of "budgeting" the calcium SR stores such that when the stores are low, the given CRU becomes much less likely to open; each CRU is assigned a uniformly distributed random value u_{rand}, which is compared to the single value returned by the CRU opening probability J_{prob} to determine whether or not the given CRU will open. (iii) The Dirac delta distribution $\delta(\mathbf{x} - \hat{\mathbf{x}})$ models each CRU as a point source for calcium release, which is defined by requiring $\delta(\mathbf{x} - \hat{\mathbf{x}}) = 0$ for all $\mathbf{x} \neq \hat{\mathbf{x}}$ and $\int_{\mathbb{R}^3} \psi(\mathbf{x})\delta(\mathbf{x} - \hat{\mathbf{x}})\, d\mathbf{x} = \psi(\hat{\mathbf{x}})$ for any continuous function $\psi(\mathbf{x})$.

The calcium signaling portion of the model consists of the equations (7)–(8). The membrane potential of the cell depends on both the cytosol calcium ion concentration and also on the cytosol potassium ion (K^+) concentration[13,14]. In our model, the ω term in (7) quantifies a dependence of V on c to complete the coupling from the chemical to the electrical systems in link ② in Figure 1 (a)[9], after c in (2) already contains several terms that depend on V to implement link ① in Figure 1 (a).

The connection between (2) and (7), link ① in Figure 1 (a), the link from the electrical system to the calcium system, comes through

$$J_{LCC} = \frac{\tau_{flux}}{2F} S\, g_{Ca}\, m_\infty(V)\,(V - V_{Ca}), \tag{18}$$

the only calcium flux term to involve voltage. Note the parameter κ in (2), which is an external scaling factor for J_{LCC} rather than an intrinsic physiological component; if the value of κ is set to 0, the connection, link ① in Figure 1 (a), is effectively switched off and the calcium dynamics are then modeled as though voltage were not involved. The surface area, S, of the cell is included in light of the fact that J_{LCC} describes the influx of calcium through L-type calcium channels (LCCs), which are present in the enclosing plasma membrane of the cell: the surface area of the cell is the surface area of the membrane.

We model the effect of the cytosol calcium concentration on the voltage by treating the calcium efflux term ($J_{m_{pump}} - J_{m_{leak}}$) as equivalent to the sodium-calcium exchanger current: In this case sodium concentration is treated reasonably as a fixed concentration and absorbed into the appropriate constants. Converting the calcium ion efflux using Faraday's constant, we are thus able to approximate a current generated by the sodium-calcium exchange as a function of simple calcium loss.

The individual components of the calcium efflux term are near-duplicates in form of the earlier J_{pump} and J_{leak} functions in (13) and (14), respectively. As J_{pump} described the removal of calcium from the cytosol and its transfer into SR stores,

$$J_{m_{pump}}(c) = V_{m_{pump}} \left(\frac{c^{n_{m_{pump}}}}{K_{m_{pump}}^{n_{m_{pump}}} + c^{n_{m_{pump}}}} \right) \tag{19}$$

describes the removal of calcium from the cytosol and its transfer to outside the cell across the membrane. The leak term J_{leak} described a gradual leak of calcium into the cytosol from the SR, while J_{CRU} described an abrupt, high-concentration (high relative to the leak) release of calcium into the cytosol from the SR. Similarly,

$$J_{m_{leak}} = J_{m_{pump}}(c_0) \tag{20}$$

describes a gradual leak of calcium into the cytosol from outside the cell via the plasma membrane, while J_{LCC} describes a sudden spike of calcium release into the cytosol via the LCCs.

3 | NUMERICAL METHOD

In order to do calculations for the CICR model, we need to solve a system of time-dependent parabolic partial differential equations (PDEs). These PDEs are coupled by several non-linear reaction and source terms. The current simulations use $n_s = 7$ physiological variables. The domain in our model is a hexahedron with isotropic CRU distribution as shown in Figure 1 (b). A typical cell has on the order of $15 \times 15 \times 31 = 6{,}975$ calcium release units (CRUs) throughout the cell. A cell is reasonably modeled by the hexahedral shape elongated in the z-direction, since the focus of CICR research is on estimating reasonable physiological parameter values. Taking a method of lines (MOL) approach to spatially discretize this model, we use the finite volume method (FVM) as the spatial discretization to ensure mass conservation on the discrete level and also to accommodate advection terms in the future, with $N = (N_x + 1)(N_y + 1)(N_z + 1)$ control volumes. Applying this to the case of the n_s PDEs results in a large system of ordinary differential equations (ODEs). A MOL discretization of a diffusion-reaction equations with second-order spatial derivatives results in a stiff ODE system. The time step size restrictions, due to the CFL condition, are considered too severe to allow for explicit time-stepping methods. This necessitates the use of a sophisticated ODE solver such as the family of implicit numerical differentiation formulas (NDFk). Our stiff ODEs, which needs to use an implicit ODE method, require the solution of a non-linear system. We use Newton's method as the non-linear solver, and at each Newton step we use the biconjugate gradient stabilized method (BiCGSTAB) or other Krylov subspace methods as the linear solver. Complete details of the numerical method can be found in Schäfer et al.[2]. Another possiblle numerical approach to a problem of reaction-diffusion type is the finite element method[15,16,17]. The problem considered in these references only needs to resolve one CRU, which also does not switch on and off, so a relatively coarse mesh and large time steps are sufficient due to the smoothness of the parabolic PDE. By contrast, our focus here is on parallel computing, needed both to resolve the large number of CRUs throughout the cell and their switching on and off, i.e., constant changes in the forcing terms throughout the simulation.

TABLE 2 Sizing study with $n_s = 7$ variables using double precision arithmetic, listing the mesh resolution $N_x \times N_y \times N_z$, the number of control volumes $N = (N_x + 1)(N_y + 1)(N_z + 1)$, the number of degrees of freedom (DOF) $n_{eq} = n_s N$, the number of time steps taken by the ODE solver up to final time $t_{fin} = 100$ ms, and the predicted and observed memory usage in GB for a one-process run.

Mesh resolution $N_x \times N_y \times N_z$	N	DOF n_{eq}	time steps	memory (GB)	
				predicted	observed
$32 \times 32 \times 128$	140,481	983,367	2667	0.125	< 1
$64 \times 64 \times 256$	1,085,825	7,600,775	3295	0.963	1.093
$128 \times 128 \times 512$	8,536,833	59,757,831	3867	7.569	8.476
$256 \times 256 \times 1024$	67,700,225	473,901,575	4470	60.024	67.103

While the form of the PDEs in (2)–(8) is customary, the thousands of point sources at the calcium release units (CRUs) in the forcing terms imply that standard codes have difficulty. This explains why our research centered around creating a special-purpose code[3,18,1,2]. Additionally, we leverage the regular 3-D shape of the domain to program all methods in matrix-free form. Table 2 demonstrates thus, how despite fully implicit time-stepping, even relatively fine meshes use only modest amounts of memory. This provides the key benefit now for the complete complex model with complete sophisticated numerical method stack, even with 7 or 8 variables, to fit into the high-performance memory of a KNL or a GPU. The table also shows how large the number of degrees of freedom (DOF) is, that is, the number of variables that have to be computed at every time step. With even modest meshes, there are millions of unknowns, and possibly hundreds of millions for fine meshes. This characterizes the numerical problem that needs to be tackled to resolve the large number of thousands of calcium release units (CRUs) in a cell.

The implementation of this model is done in C using MPI and OpenMP to parallelize computations. Parallelization is accomplished through block-distribution all large arrays to all MPI processes. We split of the mesh in the z-direction with one subdomain on each of the parallel processes. MPI commands such as `MPI_Isend` and `MPI_Irecv`, which are non-blocking point-to-point communication commands, send messages between neighboring processes. The collective command `MPI_Allreduce` is used for the computation of scalar products and norms.

4 | STATE-OF-THE-ART MULTI-CORE AND MANY-CORE PROCESSORS

Modern CPUs feature growing number of computational cores, such as, for instance, 18 cores in an Intel Xeon Gold 6140 Skylake CPU from 2017. Contrasted to these state-of-the-art multi-core CPUs with the most modern cores, many-core processors have even more cores, for instance, 68 cores in an Intel Xeon Phi 7250 Knights Landing (KNL) processor. The cores in a many-core processor typically feature a lower clock rate, e.g., 1.4 GHz vs. 2.3 GHz, but their larger number of cores with their rich connectivity in a 2D mesh network and the fact that significantly more high-performance memory is available on the chip itself offer the potential for better performance.

We use the Stampede2 cluster at the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for the KNL studies and the taki cluster in the UMBC High Performance Computing Facility (hpcf.umbc.edu) for the Skylake nodes. But our results do not depend on the specific clusters and apply to similar type of multi-core vs. many-core processors.

We use the Intel compiler version 18.0 and the Intel MPI implementation version 18.0, which is capable to provide optimized code for both types of processors in one executable. One of the key advantages of the KNL is that it is can serve as stand-alone processor (not as co-processor as a GPU) and it has an x86 architecture, therefore existing code can immediately be ported from CPU to KNL just by giving additional compile options. We want to leverage this simplicity here and therefore restrict our comparison to these two platforms; by contrast, to use GPUs as co-processor would require significant changes to the code and thus presents a hurdle for users.

4.1 | Multi-Core Processor: Intel 6140 Skylake

The taki cluster in HPCF at UMBC introduced the Intel Xeon Gold 6140 Skylake CPUs in 2018, clocked at 2.3 GHz and with 24.75 MB L3 cache. These chips contain 18 cores, totaling 36 cores on a two-socket node. Figure 2 illustrates the layout of a dual-socket Skylake node. This node has 384 GB of memory available in 12 DIMMs of 32 GB DDR4 memory, attached to each CPU via 6 memory channels per CPU. The memory of a Skylake node is larger than that available on a KNL node; see below. However, the Skylake's on-chip L3 cache of 24.75 MB is dramatically smaller than the KNL's 16 GB [GB sic!] of high-speed MCDRAM that the KNL chip has. The cluster taki in HPCF has 42 Skylake nodes, each with two 18-core Skylake CPUs. These nodes in the taki cluster are connected by a EDR (Enhanced Data Rate) InfiniBand interconnect (100 Gb/s bandwidth, 90 ns latency).

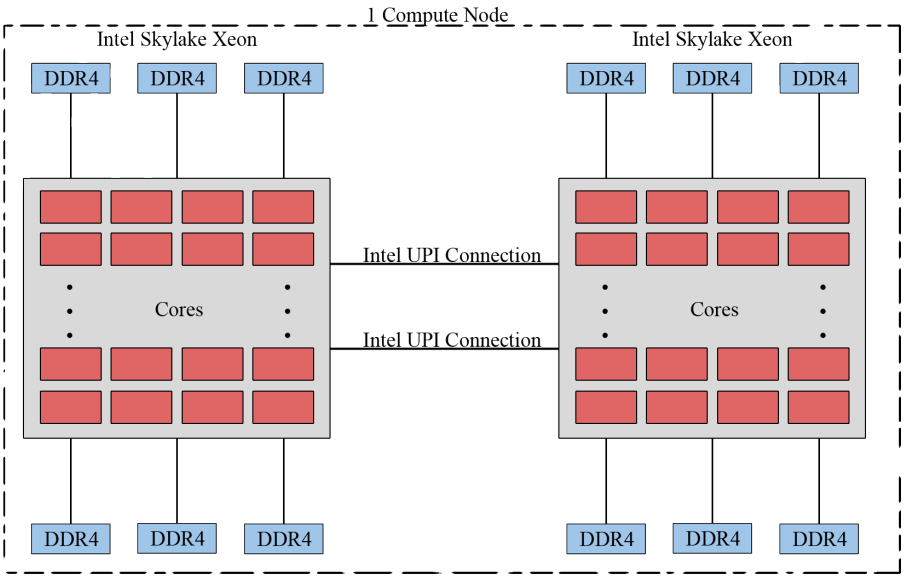


FIGURE 2 Intel Xeon Skylake schematic.

4.2 | Many-Core Processor: Intel Xeon Phi 7250 KNL

Figure 3 illustrates the layout of one Intel Xeon Phi 7250 Knights Landing (KNL) processor, whose cores are clocked at 1.4 GHz. Each KNL core is capable of 4 hardware threads. This model of the KNL on Stampede2 contains 68 cores, which are arranged in pairs of two in a so-called tile that share the connection to the two-dimensional mesh network that connects all tiles. Each core has a L1 cache, while the two cores of a tile share an L2 cache. Each box without text inside (in red color) in Figure 3 indicates one tile with two cores and their shared L2 cache in the center of the tile between the cores. Inside the KNL are 16 GB of Multi-Channel DRAM (MCDRAM), arranged in 8 blocks of 2 GB each, shown on top and bottom of Figure 3. Outside of the socket, the KNL is connected to 96 GB of DDR4 memory, arranged in 6 DIMMs of 16 GB each, shown connected to their memory controllers (MC) on the left and right of Figure 3. This MCDRAM is located directly inside the KNL chip and is up to five times faster than the DDR4 memory installed on the node because of its three-dimensional arrangement[19].

The KNL uses three distinct memory modes, which determines how the processor treats the MCDRAM; as L3 cache, RAM, or a mixture of both. The KNL also offers three distinct cluster modes that ensure all cores possess the most current data among other processes, whether from main memory or MCDRAM. The differences between these cluster modes allow users to vary the amount of explicit control of memory management. For our studies, we chose to use the 'cache' memory mode and 'quadrant' cluster mode. The Stampede2 cluster at TACC has over 4,200 KNL nodes, each with one KNL processor. All nodes in the Stampede 2 cluster are connected through an Intel Omni-Path(OPA) interconnect with speeds of 100 Gb/s.

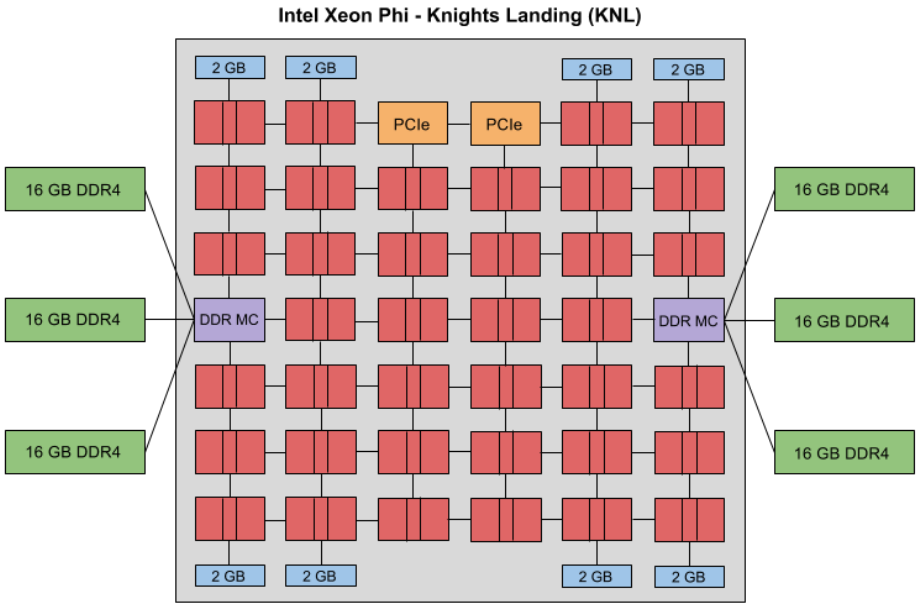


FIGURE 3 Intel Xeon Phi KNL schematic.

5 | RESULTS

This section shows the results of simulations for the seven variable model (2)–(8) with variables specified in Table 1. Section 5.1 gives an impression of the physiological output that is possible through long-time computational simulations in three spatial dimensions to a final time large enough to include several calcium waves, while Section 5.2 conducts parallel performance studies (up to a smaller reference final time that covers one wave). Both the high dimension of the space and the needed large final times lead to the need for efficient parallel computing for this problem.

5.1 | Long-Time Application Studies to $t_{fin} = 1{,}000$ ms

This section shows the results of simulations for the seven variable model to the large final time $t_{fin} = 1{,}000$ ms. Such a large final time is vital in order to capture the behavior of several waves of increasing calcium concentration to move through the cell, caused by the refractory period for the CRUs modeled as 100 ms.

The first set of plots, in Figure 4, display the locations of open calcium release units by a fat dot. The more dark dots are visible, the more CRUs are open at that specific time. More specifically, since one open CRU tends to promote neighboring CRUs to open through the diffusion of calcium, a clustering of dots indicates a region of several open CRUs. Note that we start with initial conditions in Table 1 for which there are no open CRUs and the right-hand sides of all equations in (2)–(8) are zero. Thus, these simulations represent a study in spontaneous sparking, in which some CRUs happen to open according to the probabilistic model in (15)–(17). This model embodies the effect of a higher concentration of cytosol calcium increasing the probability for a CRU to open in (17). The result of this spontaneous self-initiation is seen in the first sub-plot in Figure 4, where a number of CRUs in one region of the cell have opened by $t = 100$ ms. The study of self-initiation was the original purpose of this model of calcium induced calcium releases[6,7] and only required modest final times. The extension of the simulations to larger final times allows to study if the opening and closing of CRUs self-organizes into a sustained wave. This is seen in the following sub-plots in Figure 4. The model requires CRUs to close for 100 ms after opening for 5 ms, thus the sub-plots indicate that the original cluster of open CRUs travels in both directions through the elongated direction of the cell and roughly repeats every 100 ms. Movies of the full results including intervening times confirm this behavior.

Figure 5 shows a collection of isosurface plots for calcium concentration in the cytosol. An isosurface plot displays the surface in the three-dimensional cell, where the species concentration is equal to a critical value, stated in the caption of the figure, here 65 μM for the concentration of cytosol calcium $c(\mathbf{x}, t)$. Due to the connection with open CRUs through the effect of calcium induced calcium release, calcium concentration in the cytosol in Figure 5 is high in the same areas of the cell where the CRUs are open in Figure 4 at that time.

In order to demonstrate the capability of the model and its implementation, we show the example of one additional species, selected as the latest additional species in the model and because of its connection to the Mechanical Contraction in Figure 1 (a) as the latest addition to the model[12]. the concentration of the inactive actin-myosin cross-bridges $b_3^{(c)}(\mathbf{x}, t)$ throughout the cell in Figure 6. This species interacts with with the buffer species $b_2^{(c)}$ through $R_3^{(c)}$ in (12), which in turn reacts with the cytosol calcium $c(\mathbf{x}, t)$ through the reaction rate $R_2^{(c)}$ in (10). An increasing calcium concentration $c(\mathbf{x}, t)$ promotes the activation of the actin-myosin cross-bridges, thus the concentration of the inactive cross-bridges $b_3^{(c)}(\mathbf{x}, t)$ decreases from the high value in Table 1 at the initial time $t = 0$ ms, still visible at time $t = 100$ ms in the first subplot of Figure 6, where it is indicated by near-uniform color along the entire boundary of the domain, to lower values than the critical isolevel value of 50 μM in areas behind the calcium wave. The empty areas in the plots of $b_3^{(c)}$ indicate the concentration of the inactive cross-bridges as low, hence a high concentration of the active ones.

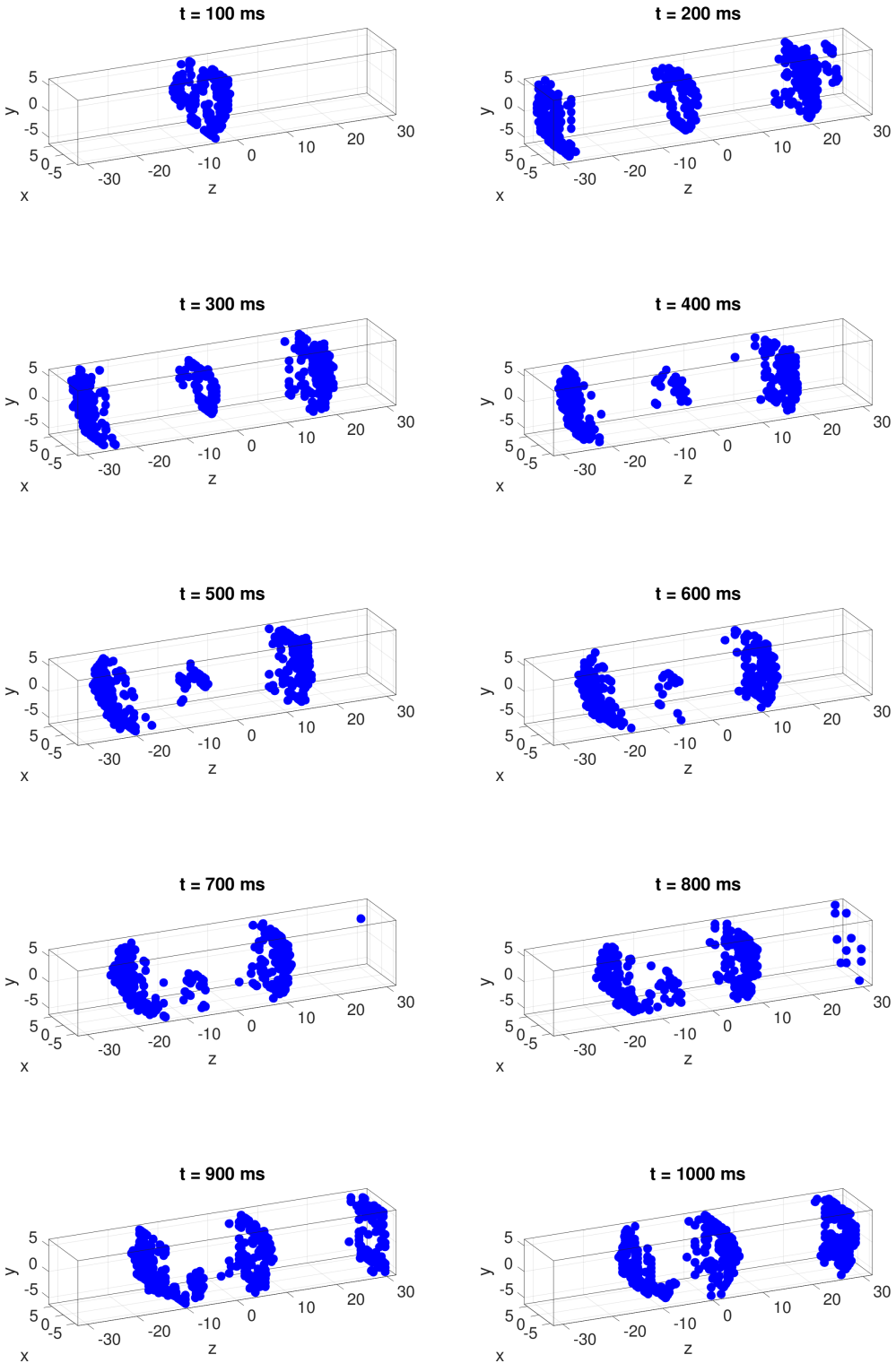


FIGURE 4 Open calcium release units throughout the cell.

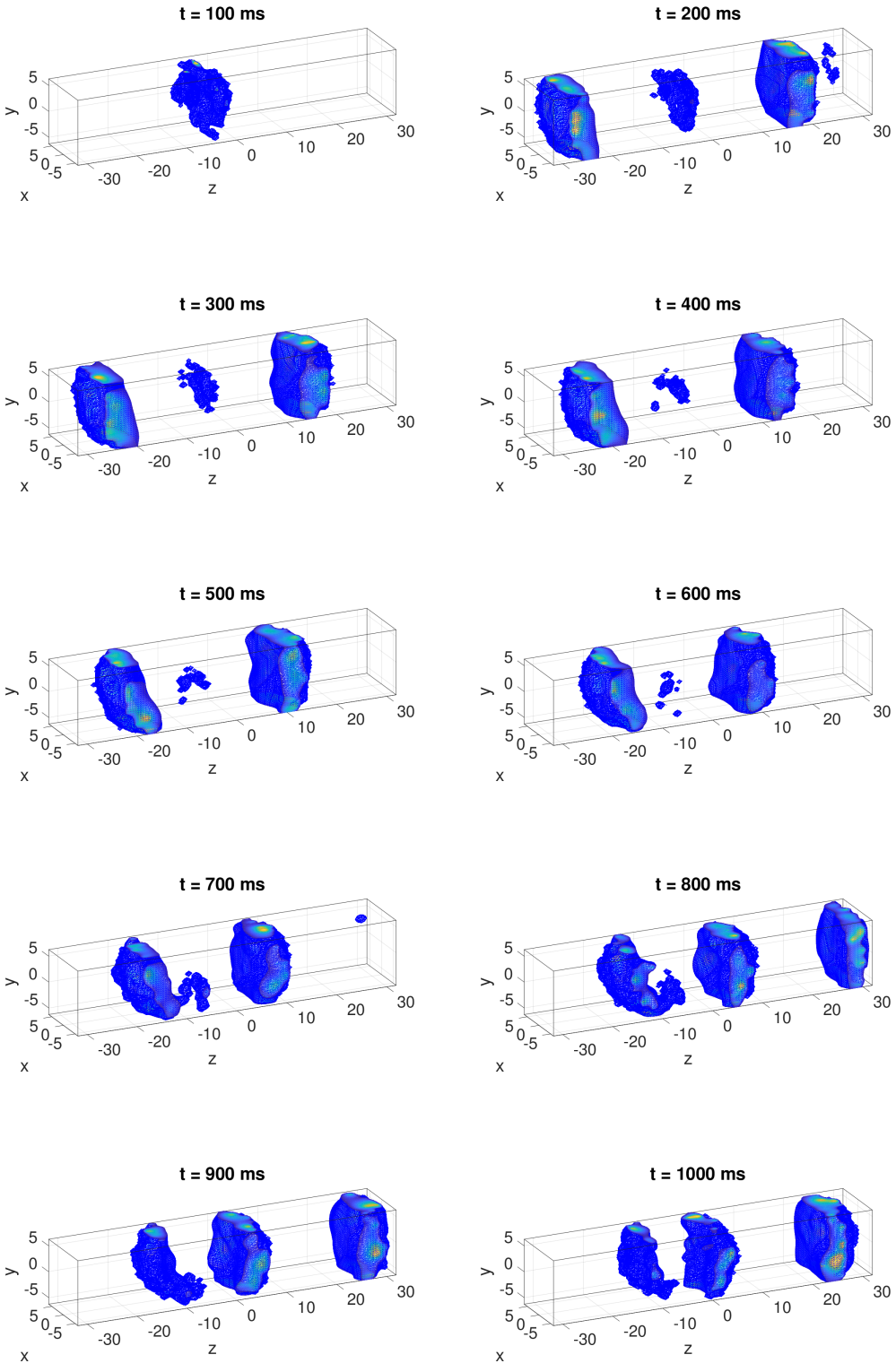


FIGURE 5 Concentration of $c(\mathbf{x}, t)$ throughout the cell with a critical value of 65 μM.

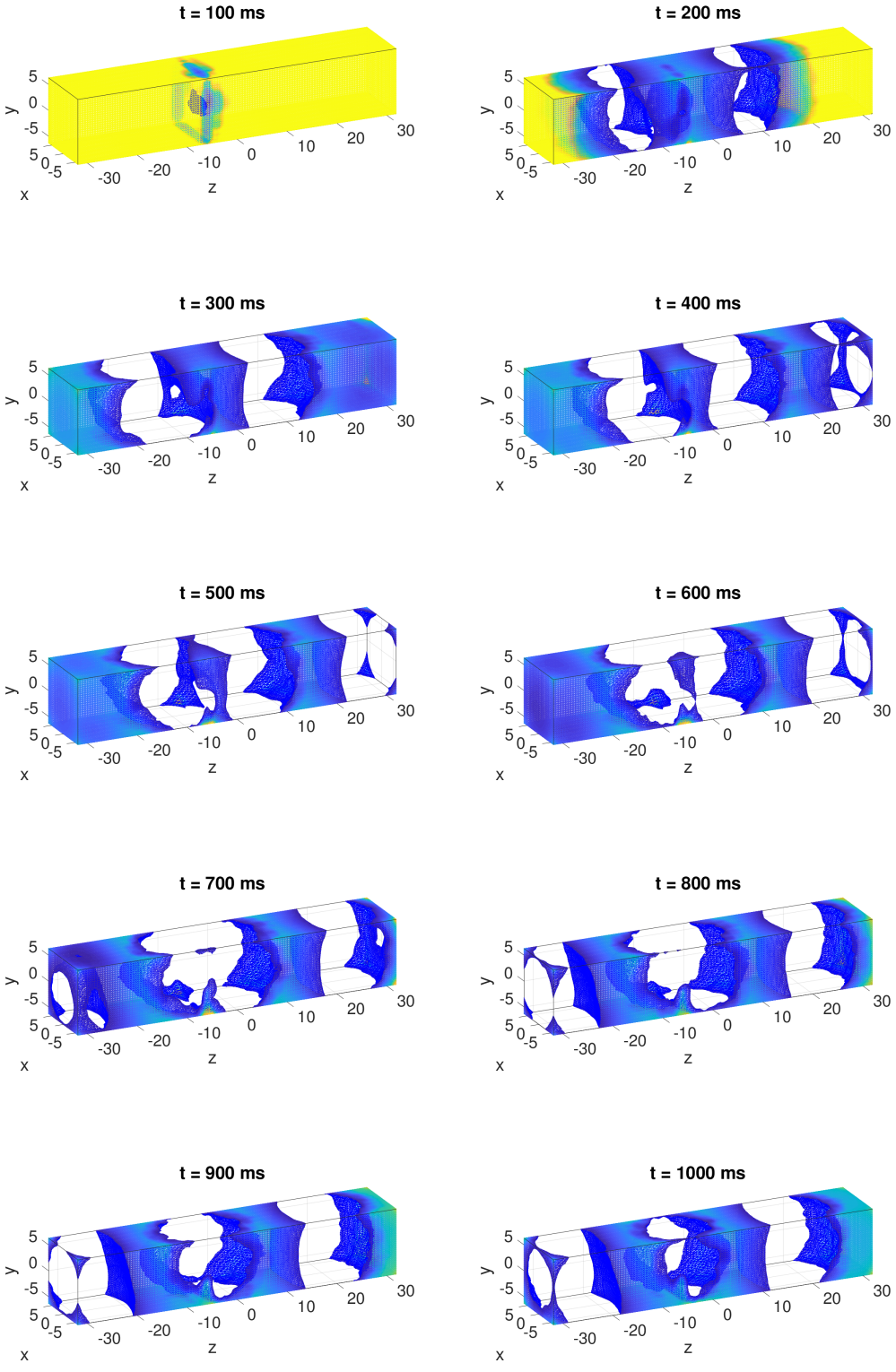


FIGURE 6 Concentration of $b_3^{(c)}(\mathbf{x}, t)$ throughout the cell with a critical value of 50 μM.

5.2 | Parallel Performance Studies to $t_{fin} = 100$ ms

The solution of PDEs of the type in (2)–(8) requires sophisticated numerical methods[4,3,18,2]. Recent developments in parallel computing architectures includes the use of graphics processing units (GPUs) as a massively parallel accelerators in general purpose computing and many-integrated-core (MIC) architectures like the Intel Xeon Phi. The second generation of the Intel Xeon Phi, code-named Knights Landing (KNL), is not limited to working as an accelerator with a CPU any more, but can be installed as a stand-alone CPU. Besides the larger number of computational cores, for instance 68 cores for the KNL, the key difference to a CPU is the significant on-chip memory in a KNL or GPU, on the order of several GB. For this application and its implementation in a special-purpose code with matrix-free linear solves, this is ideal since even relatively fine meshes fit into the memory of a KNL or a GPU; see Table 2 above. However, state-of-the-art multi-core CPUs such as the Intel Skylake CPU are available with, for instance, 18 cores, but their on-chip L3 cache is significantly smaller than the high-performance memory in the KNL chip, thus the CPU needs to rely on the memory installed on the node for all calculations.

The purpose of this section is to demonstrate the parallel scalability of the MPI code as well as to compare its performance on Skylake CPUs vs. KNLs. To avoid excessive run times for the performance studies in this subsection, we run the performance studies in this section to a final time of 100 ms, rather than the full simulation time of 1,000 ms used for the CICR model in the application subsection above.

As the mesh size N increases, the run times increase as well but the quality of the results improves with a finer mesh. Since there is a trade-off of quality with time due to finer meshes, there is a need to try and find the optimal running scheme with which we can see the best run time for the finer meshes. So, we will also vary the mesh size in order to see the full breadth of the impact of mesh density on performance time.

Table 3 collects the results of the performance studies for three meshes run on combinations of nodes from 1 to 32 by powers of 2 and processes per node from 1 to 32 by powers of 2. The table summarizes the observed wall clock time (total time to execute the code) in HH:MM:SS (hours:minutes:seconds) format. The numerical mesh in 3-D with resolution $N_x \times N_y \times N_z$ is divided onto the p parallel processes by dividing in the last dimension with N_z elements. This minimizes the size of the interfaces between subdomains that need to be communicated between neighboring process pairs. For N_z to be divided into p processes, p cannot be larger than N_z and cases with $p > N_z$ are marked as N/A in the table. For the mesh resolutions $32 \times 32 \times 128$, $64 \times 64 \times 256$, and $128 \times 128 \times 512$, a serial run on one node is feasible as basis for comparisons. For an additional mesh with $256 \times 256 \times 1{,}024$ mesh points, this is not reasonable any more, but using 32 nodes with 32 processes per node, a run is possible in a little over 3 hours (03:23:03). Recall from Table 2 that a seven variable model on the four meshes listed above requires time-stepping as well as the solutions of a system of non-linear and eventually of linear equations with nearly one million (983,367), nearly eight million (7,600,775), nearly 60 million (59,757,831), and nearly 500 million (473,901,575) variables, respectively.

Reading along the first column of a subtable, we observe that by doubling the number of processes from 1 to 2 we approximately halve the runtime from each column to the next. We observe the same improvement from 2 to 4 processes as well as from 4 to 8 processes. We also observe that by doubling the number of processes from 8 to 16 processes, there is still a significant improvement in runtime, although not the halving we observed previously. Finally, while the decrease in run time from 16 to 32 processes is small, the run times still do decrease, making the use of all available cores on a node advisable. We observe that the behavior is analogous also in all other columns for the subtables. This behavior is a typical characteristic of memory-bound code such as this. The limiting factor in performance of memory-bound code is memory access, so we would expect a bottleneck when more processes on each CPU attempt to access the memory simultaneously than the available 6 memory channels per CPU; see Figure 2.

Reading along each row of the mesh subtables, we observe that by doubling the number of nodes used, and thus also doubling the number of parallel processes, we approximately halve the runtime all the way up to 32 nodes, at least for the larger meshes. This behavior observed for increasing the number of nodes confirms the quality of the high-performance interconnect between the nodes. Also, we can see that the timings for anti-diagonals in Table 3 are about equal, that is, the run time for nodes=2 and and processes per node=1 is almost same as for nodes=1 and processes per node=2, for instance. Thus, it is advisable to use the smallest number of nodes with the largest number of processes per node.

Parallel scalability is often visually represented by plots of observed speedup and efficiency. The ideal behavior of code for a fixed problem size N using p parallel processes is that it be p times as fast as serial code. If $T_p(N)$ denotes the wall clock time for a problem of a fixed size parameterized by N using p processes, then the quantity $S_p = T_1(N)/T_p(N)$ measures the speedup of the code from 1 to p processes, whose optimal value is $S_p = p$. The efficiency $E_p = S_p/p$ characterizes in relative terms how close a run with p parallel processes is to this optimal value, for which $E_p = 1$. The behavior described here for speedup for a fixed problem size is known as strong scalability of parallel code.

TABLE 3 Parallel performance study on Skylake nodes of taki: wall clock time in HH:MM:SS for combinations of number of nodes and numbers of processes per node.

(a) Mesh resolution $N_x \times N_y \times N_z = 32 \times 32 \times 128$, system dimension 983,367						
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes
1 process per node	00:21:22	00:11:17	00:06:20	00:03:49	00:02:36	00:02:00
2 processes per node	00:11:19	00:06:20	00:03:50	00:02:37	00:02:01	00:01:43
4 processes per node	00:06:34	00:03:58	00:02:40	00:02:05	00:01:46	00:01:37
8 processes per node	00:04:13	00:02:50	00:02:09	00:01:50	00:01:41	N/A
16 processes per node	00:02:52	00:02:12	00:01:51	00:01:42	N/A	N/A
32 processes per node	00:02:27	00:02:04	00:01:53	N/A	N/A	N/A
(b) Mesh resolution $N_x \times N_y \times N_z = 64 \times 64 \times 256$, system dimension 7,600,775						
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes
1 process per node	04:33:47	02:18:40	01:09:59	00:35:40	00:19:04	00:10:27
2 processes per node	02:19:40	01:10:17	00:35:48	00:19:00	00:10:30	00:06:23
4 processes per node	01:13:49	00:37:36	00:20:15	00:11:01	00:06:44	00:04:33
8 processes per node	00:38:38	00:20:51	00:11:25	00:06:59	00:04:44	00:03:36
16 processes per node	00:21:46	00:11:54	00:07:13	00:04:47	00:03:39	N/A
32 processes per node	00:13:43	00:08:19	00:05:30	00:04:08	N/A	N/A
(c) Mesh resolution $N_x \times N_y \times N_z = 128 \times 128 \times 512$, system dimension 59,757,831						
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes
1 process per node	68:58:40	33:53:17	17:14:28	08:26:30	04:17:03	02:15:29
2 processes per node	34:43:21	17:03:16	08:30:27	04:18:29	02:14:30	01:11:13
4 processes per node	18:06:38	08:56:06	04:34:48	02:19:28	01:15:14	00:41:51
8 processes per node	09:08:44	04:41:48	02:21:23	01:19:02	00:43:30	00:26:44
16 processes per node	04:55:07	02:32:33	01:21:24	00:45:54	00:27:41	00:18:48
32 processes per node	03:00:11	01:34:39	00:53:07	00:31:36	00:21:36	N/A

Table 4 organizes the results of Table 3 in the form of a strong scalability study, that is, there is one row for each problem size, with columns for increasing number of parallel processes p. Table 4 (a) lists the raw timing data, like Table 3, but organized by numbers of parallel processes p. Tables 4 (b) and (c) show the numbers for speedup and efficiency, respectively, that will be visualized in Figures 7 (a) and (b), respectively. There are several choices for most values of p, such as for instance for $p = 4$, one could use 1 node with 4 processes per node, 2 nodes with 2 processes per node, or 4 nodes with 1 process per node. In all cases, we use the smallest number of nodes possible, with 32 processes per node for $p \geq 32$; for $p < 32$, only one node is used, with the remaining cores idle. Comparing adjacent columns in the raw timing data in Table 4 (a) confirms our previous observation that performance improvement is very good from 1 to 2 processes and from 2 to 4 processes, but not quite as good from 4 to 8 processes. The speedup numbers in Table 4 (b) help reach the same conclusions when speedup is very good for $p \leq 8$, before it degrades slowly. We observe clearly that for larger meshes, the speedup remains much better for larger p. The efficiency data in Table 4 (c) can bring out these effects more quantitatively, namely efficiency is near-optimal for $p \leq 8$, and remains very good for larger p for the larger meshes.

The plots in Figures 7 (a) and (b) visualize the numbers in Tables 4 (b) and (c), respectively. These plots do not provide new data but simply provide a graphical representation of the results in Table 4. It is customary in results for fixed problem sizes that the speedup is better for larger problems, since the increased communication time for more parallel processes does not dominate over the calculation time as quickly as it does for small problems. This is born out generally by both plots in Figure 7. Specifically, the speedup in Figure 7 (a) appears near-optimal in the beginning for each line and then slowly degrades, later for larger meshes. One would expect that the efficiency plot in Figure 7 (b) would not add much clarity, since its data are directly derived from the speedup data. But the efficiency plot can provide insight into behavior for small p, here showing that some loss of efficiency does accur there already. Beyond small p, Figure 7 (b) shows a slow degradation of efficiency, but remaining good longer for finer meshes.

TABLE 4 Parallel performance study on Skylake nodes: strong performance study for difference numbers of MPI processes.

(a) Wall clock time T_p in HH:MM:SS

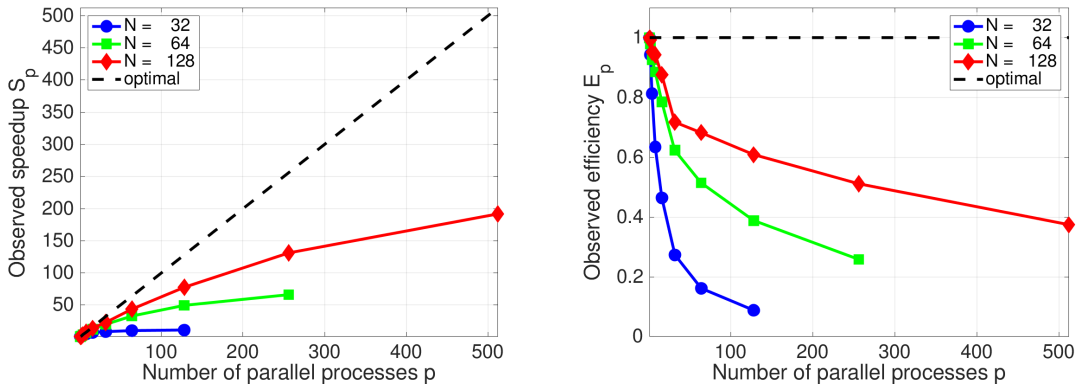
N	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$	$p=32$	$p=64$	$p=128$	$p=256$	$p=512$
32	00:21:22	00:11:19	00:06:34	00:04:13	00:02:52	00:02:27	00:02:04	00:01:53	N/A	N/A
64	04:33:47	02:19:40	01:13:49	00:38:38	00:21:46	00:13:43	00:08:19	00:05:30	00:04:08	N/A
128	68:58:40	34:43:21	18:06:38	09:08:44	04:55:07	03:00:11	01:34:39	00:53:07	00:31:36	00:21:36

(b) Observed speedup $S_p = T_1/T_p$

N	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$	$p=32$	$p=64$	$p=128$	$p=256$	$p=512$
32	1.00	1.89	3.25	5.08	7.44	8.75	10.36	11.30	N/A	N/A
64	1.00	1.96	3.71	7.09	12.58	19.97	32.94	49.71	66.18	N/A
128	1.00	1.99	3.81	7.54	14.02	22.97	43.72	77.92	130.98	191.63

(c) Observed efficiency $E_p = S_p/p$

N	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$	$p=32$	$p=64$	$p=128$	$p=256$	$p=512$
32	1.00	0.94	0.81	0.63	0.46	0.27	0.16	0.09	N/A	N/A
64	1.00	0.98	0.93	0.89	0.79	0.62	0.51	0.39	0.26	N/A
128	1.00	0.99	0.95	0.94	0.88	0.72	0.68	0.61	0.51	0.37



(a) Observed speedup S_p (b) Observed efficiency E_p

FIGURE 7 Parallel performance study on Skylake nodes: strong performance study for difference numbers of MPI processes.

Finally, Table 5 gives the performance results for using KNL nodes on Stampede2 is the same format as Table 3. Since each core in a KNL is not a very powerful processor and runs at a low clock rate, it only makes sense to use a large number of cores per node. Thus, we run immediately with 64 processes per node, leaving a few cores of the available 68 for the operating system and management of OMB-Path Architecture (OPA) traffic, as recommended in Rosales et al.[20]. We leave some cores free as is recommended in[20] for the management of OMB-Path Architecture (OPA) traffic and improves scalability. The KNL allows for up to 4 hardware threads per core, but using more than 1 proved detrimental to performance, in fact, hence we restrict the studies to single-threading. The numbers for the two meshes in Table 5 bear out that one KNL is faster than a serial job on a Skylake node, but when utilizing more cores on a Skylake node, the code is faster than on a KNL node. The speedup by doubling the number of KNL nodes is good, but runs on Skylake nodes are faster in all cases.

TABLE 5 Parallel performance study on KNL nodes of Stampede2: wall clock time in HH:MM:SS for several number of nodes.

(a) Mesh resolution $32 \times 32 \times 128$			
	1 node	2 nodes	4 nodes
64 processes per node	00:15:02	00:13:43	N/A
(b) Mesh resolution $64 \times 64 \times 256$			
	1 node	2 nodes	4 nodes
64 processes per node	00:57:45	00:39:25	00:30:05

6 | CONCLUSIONS AND FUTURE WORK

We detailed a seven variable model for the excitation-contraction coupling (ECC) occurring in the cardiomyocyte, in which calcium induced calcium release (CICR) is the mechanism through which electrical excitation is coupled with mechanical contraction through calcium signaling. The challenges of simulations for this application include the need for a fine mesh to resolve the large number of thousands of calcium release units (CRUs) as point sources combined with a need for long-time simulations to match the laboratory time scales of minutes. The simulations shown here bear out that long-time studies for modest meshes are possible up to moderate times like 1,000 ms, but it still poses challenges to reach several minutes. Higher-resolution meshes however still need large numbers of nodes in a parallel computing cluster.

This situation poses opportunities for more research into the numerical algorithms, their implementation, as well as the choice of computer hardware. Our results bring out that the second-generation Intel Xeon Phi Knights Landing (KNL), despite its promise with large on-chip memory, is not competitive with more customary CPU-based nodes, though. The results on multiple nodes with Intel Skylake CPUs show very good speedup up to 32 nodes and also demonstrate that using essentially all cores on a node is beneficial.

The seven variable model used here as application example for the broader class of systems of non-linear time-dependent parabolic PDEs also requires more work. For one thing, additional chemical species should be included. For another, larger final times are needed to take advantage of the model's ability to study long-time behavior. For these goals, far more significant numbers of studies are needed to determine reasonable ranges of parameters, so that the model shows realistic behavior. This is also an example of uncertainty quantification, since the code contains both a random number generator as well as repeated studies are needed to address uncertainty in the paramter values in the simulations. We note that an efficient parallel code is vital to facilitate larger numbers of studies, the possibility for which we demonstrated with the performance study here.

ACKNOWLEDGEMENTS

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI–1053575[21]. We acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing the HPC resources Stampede 2. The hardware in the UMBC High Performance Computing Facility (HPCF) is supported by the U.S. National Science Foundation through the MRI program (grant nos. CNS–0821258, CNS–1228778, and OAC–1726023) and the SCREMS program (grant no. DMS–0821311), with additional substantial support from the University of Maryland, Baltimore County (UMBC). See hpcf.umbc.edu for more information on HPCF and the projects using its resources. Co-author Carlos Barajas was supported by UMBC as HPCF RA.

References

1. Seidman TI, Gobbert MK, Trott DW, Kružík M. Finite Element Approximation for Time-Dependent Diffusion with Measure-Valued Source. *Numer. Math.* 2012; 122(4): 709–723.

2. Schäfer J, Huang X, Kopecz S, Birken P, Gobbert MK, Meister A. A Memory-Efficient Finite Volume Method for Advection-Diffusion-Reaction Systems with Non-Smooth Sources. *Numer. Methods Partial Differential Equations* 2015; 31(1): 143–167.

3. Gobbert MK. Long-Time Simulations on High Resolution Meshes to Model Calcium Waves in a Heart Cell. *SIAM J. Sci. Comput.* 2008; 30(6): 2922–2947.

4. Izu LT, Means SA, Shadid JN, Chen-Izu Y, Balke CW. Interplay of Ryanodine Receptor Distribution and Calcium Dynamics. *Biophys. J.* 2006; 91: 95–112.

5. Mozaffarian B, others . Heart Disease and Stroke Statistics — 2015 Update: A Report From the American Heart Association. *Circulation* 2014; 131(4): e29.

6. Izu LT, Mauban JRH, Balke CW, Wier WG. Large currents generate cardiac Ca^{2+} sparks. *Biophys. J.* 2001; 80: 88–102.

7. Izu LT, Wier WG, Balke CW. Evolution of cardiac calcium waves from stochastic calcium sparks. *Biophys. J.* 2001; 80: 103–120.

8. Alexander AM, DeNardo EK, Frazier III E, et al. Spontaneous Calcium Release in Cardiac Myocytes: Store Overload and Electrical Dynamics. *Spora: A Journal of Biomathematics* 2015; 1: 36–48.

9. Angeloff K, Barajas CA, Middleton A, et al. Examining the Effect of Introducing a Link from Electrical Excitation to Calcium Dynamics in a Cardiomyocyte. *Spora: A Journal of Biomathematics* 2016; 2: 49–73.

10. Deetz K, Foster N, Leftwich D, et al. Developing the Coupling of the Mechanical to the Electrical and Calcium Systems in a Heart Cell. Tech. Rep. HPCF–2017–15, UMBC High Performance Computing Facility, University of Maryland, Baltimore County; 2017.

11. Deetz K, Foster N, Leftwich D, et al. Examining the Electrical Excitation, Calcium Signaling, and Mechanical Contraction Cycle in a Heart Cell. *Spora: A Journal of Biomathematics* 2017; 3: 66–85.

12. Kroiz GC, Barajas C, Gobbert MK, Peercy BE. Linkages of Calcium Induced Calcium Release in a Cardiomyocyte Simulated by a System of Seven Coupled Partial Differential Equations. *Involve: A Journal of Mathematics*, submitted (2019).

13. Banyasz T, Horvath B, Jian Z, Izu LT, Chen-Izu Y. Profile of L-type Ca 2+ current and Na+/Ca 2+ exchange current during cardiac action potential in ventricular myocytes. *Heart Rhythm* 2012; 9(1): 134–142.

14. Morris C, Lecar H. Voltage oscillations in the barnacle giant muscle fiber.. *Biophys. J.* 1981; 35(1): 193.

15. Naik PA, Pardasani KR. 2D Finite-Element Analysis of Calcium Distribution in Oocytes. *Network Modeling Analysis in Health Informatics and Bioinformatics* 2018; 7: 1–11.

16. Naik PA, Pardasani KR. Finite Element Model to Study Effect of Na^+/K^+ Pump and Na^+/Ca^{2+} Exchanger on Calcium Distribution in Oocytes in Presence of Buffers. *Asian Journal of Mathematics and Statistics* 2014; 7(1): 21–28.

17. Naik PA, Pardasani KR. Three-Dimensional Finite Element Model to Study Effect of RyR Calcium Channel, ER Leak and SERCA Pump on Calcium Distribution in Oocyte Cell. *International Journal of Computational Methods* 2019; 16(1): 1–20.

18. Hanhart AL, Gobbert MK, Izu LT. A Memory-Efficient Finite Element Method for Systems of Reaction-Diffusion Equations with Non-Smooth Forcing. *J. Comput. Appl. Math.* 2004; 169(2): 431–458.

19. Sodani A, Gramunt R, Corbal J, et al. Knights Landing: Second-Generation Intel Xeon Phi Product. *IEEE Micro* 2016; 32(2): 34-46.

20. Rosales C, James D, Gómez-Iglesias A, et al. KNL Utilization Guidelines. Tech. Rep. TR–16–03, Texas Advanced Computing Center, The University of Texas at Austin; 2016.

21. Towns J, Cockerill T, Dahan M, et al. XSEDE: Accelerating Scientific Discovery. *Comput. Sci. Eng.* 2014; 16(5): 62–74.

How to cite this article: Barajas C, Gobbert MK, Kroiz GC Peercy BE.Challenges and opportunities for the simulation of calcium waves on modern multi-core and many-core parallel computing platforms. *Int J Numer Meth Biomed Engng. 2019.*