Parallel Performance Studies for a Three-Species Application Problem on the Cluster maya

Xuan Huang and Matthias K. Gobbert ({hu6,gobbert}@umbc.edu)

Department of Mathematics and Statistics, University of Maryland, Baltimore County

Technical Report HPCF–2015–8, www.umbc.edu/hpcf > Publications

Abstract

High performance parallel computing depends on the interaction of a number of factors including the processors, the architecture of the compute nodes, their interconnect network, and the numerical method, and its implication. In this note, we present performance and scalability studies on the cluster maya using an existing parallel code for a three-species application problem. This application problem requires long-time simulations on a fine mesh, thus posing a very computationally intensive problem. The speedup of run times afforded by parallel computing makes the difference between simply unacceptably long runs to obtain the results (e.g., several days) and practically feasible studies (e.g., hours). The results also support the scheduling policy implemented, since they confirm that it is beneficial to use all sixteen cores of the two eight-core processors on each node simultaneously.

1 Introduction

The UMBC High Performance Computing Facility (HPCF) is the community-based, interdisciplinary core facility for scientific computing and research on parallel algorithms at UMBC. Started in 2008 by more than 20 researchers from ten academic departments and research centers from all three colleges, it is supported by faculty contributions, federal grants, and the UMBC administration. The facility is open to UMBC researchers at no charge. Researchers can contribute funding for long-term priority access. System administration is provided by the UMBC Division of Information Technology, and users have access to consulting support provided by dedicated full-time graduate assistants. See www.umbc.edu/hpcf for more information on HPCF and the projects using its resources.

Released in Summer 2014, the current machine in HPCF is the 240-node distributed-memory cluster maya. The newest components of the cluster are the 72 nodes in maya 2013 with two eight-core 2.6 GHz Intel E5-2650v2 Ivy Bridge CPUs and 64 GB memory that include 19 hybrid nodes with two state-of-the-art NVIDIA K20 GPUs (graphics processing units) designed for scientific computing and 19 hybrid nodes with two cutting-edge 60-core Intel Phi 5110P accelerators. These new nodes are connected along with the 84 nodes in maya 2009 with two quad-core 2.6 GHz Intel Nehalem X5550 CPUs and 24 GB memory by a high-speed quad-data rate (QDR) InfiniBand network for research on parallel algorithms. The remaining 84 nodes in maya 2010 with two quad-core 2.8 GHz Intel Nehalem X5560 CPUs and 24 GB memory are designed for fastest number crunching and connected by a dual-data rate (DDR) InfiniBand network. All nodes are connected via InfiniBand to a central storage of more than 750 TB.

This report is an update to the technical report [12], which considered an earlier version of the application problem on the previous cluster tara. The studies in this reports use default Intel C compiler version 14.0 (compiler options `-std=c99 -Wall -O3`) with Intel MPI version 4.1. All results in this report use dedicated nodes with unused cores idling using the `--exclusive` option in the SLURM submission script.

An important, practical approach to testing the real-life performance of a computer is to perform studies using reliable high performance code that is already being used in production. Performance tests of this nature not only provide a tool for gauging the effectiveness of a specific hardware setup, but they can also provide guidance to selecting a particular usage policy for clusters as well as give concrete experience in the expected length of production runs on the specific cluster. This note is part of a sequence of performance studies conducted on the cluster maya in HPCF. It was shown in [8] that an elliptic test problem given by the stationary Poisson equation, whose code uses a parallel, matrix-free implementation of the conjugate gradient (CG) linear solver, provides an excellent test problem since it tests two important types of parallel communications, namely collective communications involving all participating parallel processes and point-to-point communications between certain pairs of processes. The report [2] extends the stationary Poisson equation to a time-dependent parabolic test problem given by one scalar, time-dependent, linear reaction-diffusion equation. The code for this problem involves implicit time-stepping with a linear solve using CG at every time step, and thus the linear solver remains the key challenge for the parallel interconnect; since the test problem is linear, no non-linear Newton solver is used. This note extends the time-dependent parabolic test problem to a system of three non-linear advection-reaction-diffusion equations for the application problem of simulating calcium induced calcium release (CICR) in a heart cell [1, 3, 9]. This non-linear three-species application problem provides a substantially more computationally intensive test of the cluster, since

Table 1.1: Wall clock time in HH:MM:SS of the CICR problem solved with first order FVM on maya 2013 by number of nodes and processes per node. Mesh resolution $N_x \times N_y \times N_z = 128 \times 128 \times 512$, DOF = 25,610,499. ET indicates "excessive time required" (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$.

	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	ET	ET	69:15:37	34:51:02	17:31:44	08:59:06	04:49:17
2 processes per node	ET	69:46:29	35:16:03	17:45:06	09:00:47	04:47:14	02:43:04
4 processes per node	72:31:51	36:34:34	18:36:29	09:32:04	05:01:44	02:50:34	01:46:47
8 processes per node	42:01:27	26:23:03	11:03:41	05:46:44	03:09:23	01:56:47	01:23:57
16 processes per node	26:53:37	13:56:38	07:21:17	03:54:47	02:17:48	01:40:35	N/A

the model involves additional terms, the solution requires many more time steps than the test problem, it involves non-linearities requiring a Newton solver at every time step, and BiCGSTAB is used as linear solver inside every Newton iteration. Section 2 provides a brief introduction to the three-species application problem and the numerical algorithm used.

Table 1.1 contains an excerpt of the performance results reported in Table 3.1 of Section 3 for the studies on the newest portion of the maya cluster, referred to as maya 2013. Table 1.1 reports the observed wall clock time in HH:MM:SS for the highest mesh resolution 128 × 128 × 512 which results in a system of over 25 million equations to be solved at every time step. Wall clock times are given for all possible combinations of numbers of nodes and processes per node (that are powers of 2), that is, for 1, 2, 4, 8, 16, 32, and 64 nodes and 1, 2, 4, 8, and 16 processes per node. The "ET" indicates "excessive time required" (more than 5 days), the "N/A" indicates that the case is not feasible due to $p > (N_z + 1)$, where $N_z + 1$ is the number of finite volume cells on the z-direction for spatial mesh resolution of $N_x \times N_y \times N_z$. We observe good scalability while increasing the number of nodes or increasing the number of processes per node. Moreover, we observe that the serial run takes more than 5 days, while the run using either 32 or 64 nodes on maya 2013 can take less than 2 hours. These results demonstrate the power of parallel computing, since jobs require excessive time in serial can be achieved within hours using parallel computing.

The previous reports in this sequence of performance studies, [8] and [2], used significantly finer meshes resulting in larger systems of linear equations than this report. These reports refined the mesh until running out of memory and thus demonstrated one key advantage of parallel computing: Larger problems can be solved by pooling the memory from several compute nodes. By contrast, the problem in this report is an actual application problem with physical effects and modeling requirements that necessarily require many more time steps, as discussed in Section 2. Additionally, the non-linearities in the terms require more time steps per unit of time simulated, and each time step is more expensive due to the non-linearities, the larger number of terms that need to be evaluated, and the three species. The long run times for the present problem thus restrict the mesh resolutions that can be computed within reasonable amount of time. For the present problem, even the finest mesh possible within reasonable run times, $128 \times 128 \times 512$, does not put a strain on the memory of a node of the given cluster Thus, this combination of facts brings out a key advantage of parallel computing: For efficient implementations of appropriate algorithms, problems can be solved significantly faster by pooling the processing power of several compute nodes.

The remainder of this report is organized as follows: Section 2 details the application problem and summarizes the numerical methods used. Section 3 contains the complete parallel performance studies on maya 2013, from which Table 1.1 was excerpted. Section 4 contains the parallel performance studies on maya 2010. Section 5 contains the performance studies on maya 2009 and is an update of results from [12]. Finally, Section 6 provides a historical comparison of performance of maya and the previous cluster tara in HPCF. The results for maya 2013 are already contained in the report [4].

Acknowledgments

The hardware used in the computational studies is part of the UMBC High Performance Computing Facility (HPCF). The facility is supported by the U.S. National Science Foundation through the MRI program (grant nos. CNS–0821258 and CNS–1228778) and the SCREMS program (grant no. DMS–0821311), with additional substantial support from the University of Maryland, Baltimore County (UMBC). See www.umbc.edu/hpcf for more information on HPCF and the projects using its resources. Xuan Huang additionally acknowledges financial support as HPCF RA.

2 Three-Species Application Problem and Numerical Method

The three-species application problem models the calcium induced calcium release (CICR) on the scale of one heart cell. Calcium ions enter into the cell at release units distributed throughout the cell and then diffuse. At each release unit, the probability for calcium to be released increases along with the concentration of calcium, thus creating a feedback loop of waves re-generating themselves repeatedly. An accurate model of such waves is useful since they are part of the normal functioning of the heart, but can also trigger abnormal arrhythmia. This model requires simulations on the time scale of several repeated waves and on the spatial scale of the entire cell. This requires long-time studies on spatial meshes that need to have a high resolution to resolve the positions of the calcium release units throughout the entire cell. This CICR model was originally introduced in [5, 7], extended in [6], and its numerics discussed in [1, 3, 9, 10]. The problem can be modeled by a system of coupled, non-linear, time-dependent advection-diffusion-reaction equations

$$u_t^{(i)} - \nabla \cdot \left(D^{(i)} \nabla u^{(i)}\right) + \beta^{(i)} \cdot \left(\nabla u^{(i)}\right) = q^{(i)}(u^{(i)}, \dots, u^{(n_s)}, \mathbf{x}, t), \quad i = 1, \dots, n_s, \tag{2.1}$$

of n_s species with $u^{(i)} = u^{(i)}(\mathbf{x}, t)$ representing functions of space $\mathbf{x} \in \Omega \subset \mathbb{R}^3$ and time $0 \le t \le t_{\text{fin}}$. The diffusivity matrix $D^{(i)} = \operatorname{diag}(D_{11}^{(i)}, D_{22}^{(i)}, D_{33}^{(i)}) \in \mathbb{R}^{3\times3}$ consists of positive diagonal entries, which are assumed to dominate the scale of the advection velocity vectors $\beta^{(i)} \in \mathbb{R}^3$, so that the system is always of parabolic type. The model uses no-flux boundary conditions

$$\mathbf{n} \cdot (D_i(\mathbf{x}) \nabla u^{(i)}) = 0 \quad \text{for } x \in \partial\Omega,\ 0 < t \le t_{\text{fin}}, \tag{2.2}$$

and has given initial conditions

$$u^{(i)}(\mathbf{x}, 0) = u_{\text{ini}}^{(i)}(\mathbf{x}) \quad \text{for } x \in \Omega,\ t = 0. \tag{2.3}$$

Our consideration of this problem is inspired by the need to simulate calcium waves in one heart cell. We consider the rectangular spatial domain

$$\Omega = (-6.4\ \mu\text{m}, 6.4\ \mu\text{m}) \times (-6.4\ \mu\text{m}, 6.4\ \mu\text{m}) \times (-32.0\ \mu\text{m}, 32.0\ \mu\text{m}) \subset \mathbb{R}^3$$

that captures the essential size and elongated shape of a heart cell. The general system (2.1) consists of $n_s = 3$ equations corresponding to calcium ($i = 1$), an endogenous calcium buffer ($i = 2$), and a fluorescent indicator dye ($i = 3$). A complete list of the model's parameter values is given in Table 2.1.

The right-hand side $q^{(i)}$ of the system (2.1) is written in a way that distinguishes the different dependencies and effects as

$$q^{(i)}(u^{(i)}, \dots, u^{(n_s)}, \mathbf{x}, t) = f^{(i)}(\mathbf{x}, t) + r^{(i)}(u^{(1)}, \dots, u^{(n_s)}) + s^{(i)}(u^{(i)}, \mathbf{x}, t), \quad i = 1, \dots, n_s. \tag{2.4}$$

We describe terms on the right-hand side (2.4) as follows:

The linear term $f^{(i)} = f^{(i)}(\mathbf{x}, t)$ allows for the linear parabolic test problem that is used in [2]; here we set $f^{(i)} \equiv 0$ for all i.

The reaction terms

$$r^{(i)}(u^{(1)}, \dots, u^{(n_s)}) := \begin{cases} \displaystyle\sum_{j=2}^{n_s} R^{(j)}(u^{(1)}, u^{(j)}), & \text{for } i = 1, \\ R^{(i)}(u^{(1)}, u^{(i)}), & \text{for } i = 2, \dots, n_s, \end{cases} \tag{2.5}$$

where the reaction rates are given by

$$R^{(i)} = -k_i^+ u^{(1)} u^{(i)} + k_i^- \left(\overline{u}_i - u^{(i)}\right) \qquad \text{for } i = 2, \dots, n_s, \tag{2.6}$$

are modeled as autonomous non-linear functions of the different species and couple the equations in the general system (2.1).

In the present model, the term $s^{(i)}(u^{(i)}, \mathbf{x}, t)$ applies only to the calcium species $i = 1$, which is implemented using the Kronecker delta function δ_{i1} in the definition

$$s^{(i)}(u^{(i)}, \mathbf{x}, t) = \left(- J_{\text{pump}}(u^{(1)}) + J_{\text{leak}} + J_{\text{SR}}(u^{(1)}, \mathbf{x}, t)\right) \delta_{i1}, \quad i = 1, \dots, n_s, \tag{2.7}$$

The key term of the model J_{SR} houses the stochastic aspect of the model, since the calcium release unites (CRUs) which are arranged discretely on a three-dimensional lattice each have a probability of opening depending on the concentration of calcium present at that site. This process is explained through the equation

$$J_{\mathrm{SR}}(u^{(1)}, \mathbf{x}, t) = \sum_{\hat{\mathbf{x}} \in \Omega_s} g \, S_{\hat{\mathbf{x}}}(u^{(1)}, t) \, \delta(\mathbf{x} - \hat{\mathbf{x}}), \tag{2.8}$$

The equation models the superposition of calcium injection into the cell at CRUs, which are modeled as point sources at all $\hat{\mathbf{x}}$ in the set of CRU locations Ω_s. The Dirac delta distribution $\delta(\mathbf{x} - \hat{\mathbf{x}})$ together with the constant flux density g models a point source at a CRU located at $\hat{\mathbf{x}} \in \Omega_s$. $S_{\hat{\mathbf{x}}}$ is an indicator function, its value is either 1 or 0 indicating the CRU at $\hat{\mathbf{x}}$ is open or closed. The value of $S_{\hat{\mathbf{x}}}$ is determined by comparing a uniform random number to the value of the probability function

$$J_{\mathrm{prob}}(u^{(1)}) = \frac{P_{\max} \, (u^{(1)})^{n_{\mathrm{prob}}}}{(K_{\mathrm{prob}})^{n_{\mathrm{prob}}} + (u^{(1)})^{n_{\mathrm{prob}}}}. \tag{2.9}$$

When the value of the probability function is higher than the random number, then the CRU switches on by setting $S_{\hat{\mathbf{x}}} = 1$, otherwise it remains closed by $S_{\hat{\mathbf{x}}} = 0$. When the CRU is open, it stays open for 5 ms, then it remains closed for 100 ms. The term $s^{(i)}(u^{(i)}, \mathbf{x}, t)$ also houses the non-linear drain term

$$J_{\mathrm{pump}}(u^{(1)}) = \frac{V_{pump}(u^{(1)})^{n_{pump}}}{(K_{pump})^{n_{pump}} + (u^{(1)})^{n_{pump}}}$$

and the constant balance term J_{leak}. By design, these terms balance out as $J_{\mathrm{leak}} = J_{\mathrm{pump}}(0.1) \equiv const.$ for the calcium concentration at basal level 0.1 μM.

The spatial discretization of the three-species application problem with the finite volume method results in a large system of ordinary differential equations (ODEs). This ODE system is solved by the family of numerical differentiation formulas with automatic time step and method order control [11]. Since these ODE solvers are fully implicit, it is necessary to solve a fully coupled non-linear system of equations at every time step. The Newton method with an analytically supplied Jacobian is used as non-linear solver. The linear solver makes use of the iterative BiCGSTAB method with matrix-free matrix-vector multiplies. Table 2.2 summarizes several key parameters of the numerical method and its implementation. The first three columns show the spatial mesh resolution of $N_x \times N_y \times N_z$, the number of mesh points as well as finite volume cells $N = (N_x+1)(N_y+1)(N_z+1)$, and their associated numbers of unknowns $n_s N$ for the n_s species that need to be computed at every time step, commonly referred to as degrees of freedom (DOF). The following column lists the number of time steps taken by the ODE solver, which are significant and which increase with finer resolutions. The final two columns list the memory usage in GB, both predicted by counting variables in the algorithm and by observation provided in a memory log file produced from the performance run. We notice that even the finest resolution fits comfortably in the memory of one node of the cluster used.

The code used to perform the parallel computations is an extension of the one described in [1], which uses MPI for parallel communications. To compute the matrix-vector products needed in the Krylov subspace method, communication between neighboring processes is required. Non-blocking communications using `MPI_Isend` and `MPI_Irecv` are used for neighboring process communications. Furthermore, `MPI_Allreduce` is used for inner products and norm calculation. In the parallel implementation, all data are split in the z-direction such that the $(N_z + 1)$ mesh points are block-distributed to the p parallel processes. The division of the domain Ω into p subdomains in the long z-direction (instead of the shorter x- or y-directions) minimizes the amount of data that neighboring processes need to exchange across the subdomain interfaces. Since each process must have at least one mesh point in the z-direction, the number of processes p must not be larger than the number of $(N_z + 1)$ mesh points. In other words, combinations of p and N_z with $p > (N_z + 1)$ are not feasible.

Table 2.1: Table of parameters for the CICR model.

Parameter	Description	Values/Units
t	Time	ms
$\mathbf{x}$	Position	μm
u^i	Concentration	μM
Ω	Rectangular domain in μm	$(-6.4, 6.4) \times (-6.4, 6.4) \times (-32.0, 32.0)$
$D^{(1)}$	Calcium diffusion coefficient	$\text{diag}(0.15, 0.15, 0.30)$ μm^2/ ms
$D^{(2)}$	Mobile buffer diffusion coefficient	$\text{diag}(0.01, 0.01, 0.02)$ μm^2/ ms
$D^{(3)}$	Stationary buffer diffusion coefficient	$\text{diag}(0.00, 0.00, 0.00)$ μm^2/ ms
$\beta^{(i)}$	Advection velocity	μm / ms
$u^{(1)}_{\text{ini}}$	Initial calcium concentration	0.1 μM
$u^{(2)}_{\text{ini}}$	Initial mobile buffer concentration	45.9184 μM
$u^{(3)}_{\text{ini}}$	Initial stationary buffer concentration	111.8182 μM
Δx_s	CRU spacing in x-direction	0.8 μm
Δy_s	CRU spacing in y-direction	0.8 μm
Δz_s	CRU spacing in z-direction	0.2 μm
g	Flux density distribution	110.0 μM μm^3 / ms
P_{max}	Maximum probability rate	0.3 / ms
K_{prob}	Probability sensitivity	0.2 μM
n_{prob}	Probability Hill coefficient	4.0
Δt_s	CRU time step	1.0 ms
t_{open}	CRU opening time	5.0 ms
t_{closed}	CRU refractory period	100 ms
k_2^+	Forward reaction rate	0.08 / (μM ms)
k_2^-	Backward reaction rate	0.09 / ms
$\bar{u}_2$	Total of bound and unbound indicator	50.0 μM
k_3^+	Forward reaction rate	0.10 / (μM ms)
k_3^-	Backward reaction rate	0.10 / ms
$\bar{u}_3$	Total bound and unbound buffer	123.0 μM
V_{pump}	Maximum pump strength	4.0 μM / ms
K_{pump}	Pump sensitivity	0.184 μM
n_{pump}	Pump Hill coefficient	4
J_{leak}	Leak term	0.320968365152510 μM / ms

Table 2.2: Sizing study listing the mesh resolution $N_x \times N_y \times N_z$, the number of mesh points $N = (N_x + 1) \times (N_y + 1) \times (N_z + 1)$, the number of degrees of freedom (DOF $= n_s N$), the number of time steps taken by the ODE solver, and the predicted and observed memory usage in MB for a one-process run.

$N_x \times N_y \times N_z$	N	DOF	number of	memory usage (GB)	
			time steps	predicted	observed
$16 \times 16 \times 64$	18,785	56,355	34,877	0.01	0.03
$32 \times 32 \times 128$	140,481	421,443	58,416	0.05	0.08
$64 \times 64 \times 256$	1,085,825	3,257,475	73,123	0.41	0.48
$128 \times 128 \times 512$	8,536,833	25,610,499	89,088	3.24	3.68

3 Performance Studies on maya 2013

This section describes the parallel performance studies for the solution of the CICR problem on the 2013 portion of maya. These newest components are the 72 nodes with two eight-core 2.6 GHz Intel E5-2650v2 Ivy Bridge CPUs and 64 GB memory, connected by a quad-data rate (QDR) InfiniBand interconnect.

Table 3.1 collects the results of the performance study by number of nodes and processes per node. The table summarizes the observed wall clock time (total time to execute the code) in HH:MM:SS (hours:minutes:seconds) format. In situations where wall clock times are not obtained, ET indicates "excessive time required" (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$, where $N_z + 1$ is the number of finite volume cells on the z-direction for spatial mesh resolution of $N_x \times N_y \times N_z$.

We first discuss Table 3.1 (c) with mesh resolution of $64 \times 64 \times 256$ in detail as example, since this sub-table has all data possible. Reading along the first column of this sub-table, 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. We also observe that by doubling the number of processes from 4 to 8 and from 8 to 16 there are still significant improvement in runtime, although not the halving we observed previously. This is better than in the study in [8], where only small improvements in runtime are observed by doubling the number of processes from 8 to 16. This indicates our application problem is not heavily memory bound as the test problem in [8].

Table 3.1 (d) reports the observed wall clock time in HH:MM:SS for the highest mesh resolution $128 \times 128 \times 512$ which results in a system of over 25 million equations to be solved at every time step. Wall clock times are given for all possible combinations of numbers of nodes and processes per node (that are powers of 2), that is, for 1, 2, 4, 8, 16, 32, and 64 nodes and 1, 2, 4, 8, and 16 processes per node. We observe good scalability while increasing the number of nodes or increasing the number of processes per node. Moreover, we observe that the serial run takes more than 5 days, while the run using either 32 or 64 nodes on maya 2013 can take less than 2 hours. These results demonstrate the power of parallel computing, since jobs require excessive time in serial can be achieved within hours using parallel computing.

Table 3.2 collects the results of the performance study by number of processes. Each row lists the results for one problem size. Each column corresponds to the number of parallel processes p used in the run. Data is based on 16 processes per node, except for the cases $p = 1, 2, 4, 8$, where not all of the 16 cores of one node are utilized. This table is intended to demonstrate strong scalability on maya 2013, which is also one key motivation for parallel computing: The run times for a problem of a given, fixed size can be potentially dramatically reduced by spreading the work across a group of parallel processes. More precisely, the ideal behavior of code for a fixed problem size using p parallel processes is that it be p times as fast. If $T_p(N)$ denotes the wall clock time for a problem of a fixed size parametrized 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 $S_p = p$; for the finest resolution, where data are only available starting with $p = 4$, this definition is extended by the formula $S_p = 4T_4(N)/T_p(N)$. 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 $E_p = 1$.

Table 3.2 (b) shows the speedup observed. The speedup S_p is increasing significantly as we increase the number of processes. However, the ratio over the optimal value of speedup p seems to decrease as we increase the number of processes. We also observe that the speedup is better for larger problems. Table 3.2 (c) shows the observed efficiency E_p. The primary decrease of efficiency is between $p = 8$ and $p = 16$, similar to studies in [8] but not as severe. This suggests the bottle neck of CPU memory channels we observed in [8] may still be affecting the scalability on the CICR problem. The fundamental reason for the speedup and efficiency to trail off is that simply too little work is performed on each process. Due to the one-dimensional split in the z-direction into as many subdomains as parallel processes p, eventually only one or two x-y-planes of data are located on each process. This is not enough calculation work to justify the cost of communicating between the processes. In effect, this leads to a recommendation how many nodes to use for a particular $N_x \times N_y \times N_z$ mesh, with more nodes being justifiable for larger meshes.

The customary graphical representations of speedup and efficiency are presented in Figures 3.1 (a) and (b), respectively. Figure 3.1 (a) shows the speedup pattern as we observed in Table 3.2 (b) but more intuitively. The efficiency plotted in Figure 3.1 (b) is directly derived from the speedup, but the plot is still useful because it details interesting features for small values of p that are hard to discern in the speedup plot. Here, we notice the consistency of most results for small p.

Table 3.1: Performance study of the CICR problem solved with first order FVM on maya 2013 by number of nodes and processes per node. ET indicates “excessive time required” (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$.

(a) Mesh resolution $N_x \times N_y \times N_z = 16 \times 16 \times 64$, DOF = 56,355							
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	00:12:38	00:05:55	00:03:23	00:02:04	00:01:31	00:01:20	00:01:21
2 processes per node	00:07:23	00:03:13	00:02:00	00:01:27	00:01:25	00:01:19	N/A
4 processes per node	00:04:41	00:02:03	00:01:34	00:01:26	00:01:37	N/A	N/A
8 processes per node	00:03:19	00:01:32	00:01:32	00:01:46	N/A	N/A	N/A
16 processes per node	00:02:11	00:01:34	00:02:43	N/A	N/A	N/A	N/A
(b) Mesh resolution $N_x \times N_y \times N_z = 32 \times 32 \times 128$, DOF = 421,443							
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	02:22:19	01:10:46	00:36:09	00:19:17	00:10:54	00:06:37	00:04:56
2 processes per node	01:11:43	00:35:48	00:19:13	00:10:45	00:06:47	00:04:48	00:04:50
4 processes per node	00:37:52	00:19:27	00:11:07	00:06:59	00:05:18	00:04:41	N/A
8 processes per node	00:21:35	00:11:28	00:07:24	00:05:41	00:05:36	N/A	N/A
16 processes per node	00:12:58	00:07:24	00:06:30	00:07:26	N/A	N/A	N/A
(c) Mesh resolution $N_x \times N_y \times N_z = 64 \times 64 \times 256$, DOF = 3,257,475							
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	25:02:01	12:25:01	06:10:14	03:07:33	01:37:56	00:53:17	00:31:46
2 processes per node	12:25:07	06:11:38	03:08:20	01:37:41	00:52:27	00:30:36	00:20:05
4 processes per node	06:32:39	03:16:02	01:41:55	00:55:03	00:31:50	00:21:03	00:16:10
8 processes per node	03:52:48	01:53:44	01:00:20	00:34:24	00:22:30	00:18:33	N/A
16 processes per node	02:25:55	01:10:26	00:39:04	00:25:46	00:21:29	N/A	N/A
(d) Mesh resolution $N_x \times N_y \times N_z = 128 \times 128 \times 512$, DOF = 25,610,499							
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	ET	ET	69:15:37	34:51:02	17:31:44	08:59:06	04:49:17
2 processes per node	ET	69:46:29	35:16:03	17:45:06	09:00:47	04:47:14	02:43:04
4 processes per node	72:31:51	36:34:34	18:36:29	09:32:04	05:01:44	02:50:34	01:46:47
8 processes per node	42:01:27	26:23:03	11:03:41	05:46:44	03:09:23	01:56:47	01:23:57
16 processes per node	26:53:37	13:56:38	07:21:17	03:54:47	02:17:48	01:40:35	N/A

Table 3.2: Performance study of the CICR problem solved with first order FVM on maya 2013 by number of processes. Data based on 16 processes per node, except for the cases $p = 1, 2, 4, 8$. Mesh 1 represents $16 \times 16 \times 64$, Mesh 2 represents $32 \times 32 \times 128$, Mesh 3 represents $64 \times 64 \times 256$, Mesh 4 represents $128 \times 128 \times 512$. ET indicates "excessive time required" (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$. For the $128 \times 128 \times 512$ mesh, we use the modified definition $S_p = 4T_4(N)/T_p(N)$.

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

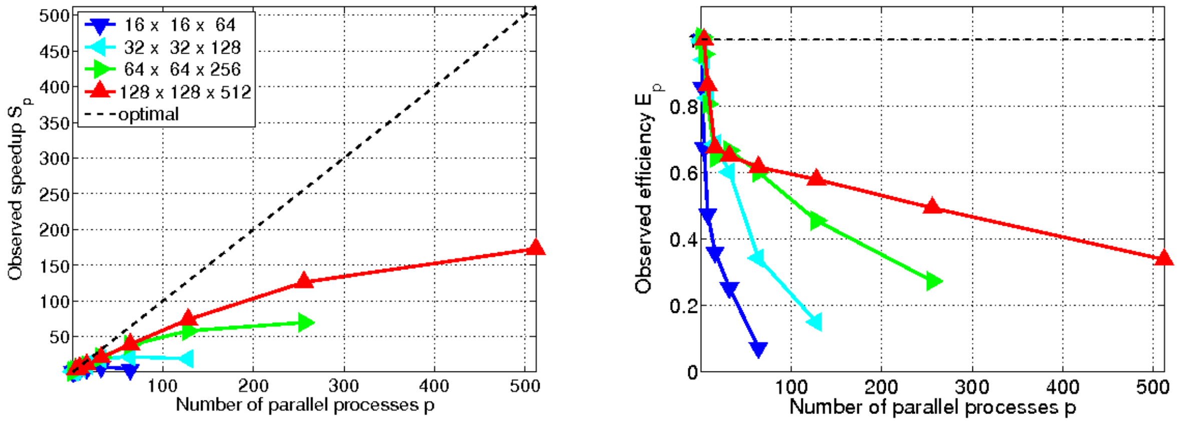
Mesh	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$	$p = 128$	$p = 256$	$p = 512$
1	00:12:38	00:07:23	00:04:41	00:03:19	00:02:11	00:01:34	00:02:43	N/A	N/A	N/A
2	02:22:19	01:11:43	00:37:52	00:21:35	00:12:58	00:07:24	00:06:30	00:07:26	N/A	N/A
3	25:02:01	12:25:07	06:32:39	03:52:48	02:25:55	01:10:26	00:39:04	00:25:46	00:21:29	N/A
4	ET	ET	72:31:51	42:01:27	26:53:37	13:56:38	07:21:17	03:54:47	02:17:48	01:40:35

(b) Observed speedup S_P

Mesh	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$	$p = 128$	$p = 256$	$p = 512$
1	1.00	1.71	2.70	3.81	5.77	8.09	4.65	N/A	N/A	N/A
2	1.00	1.98	3.76	6.60	10.97	19.25	21.92	19.16	N/A	N/A
3	1.00	2.02	3.83	6.45	10.29	21.33	38.45	58.28	69.91	N/A
4	ET	ET	4.00	6.90	10.79	20.81	39.45	74.14	126.32	173.06

(c) Observed efficiency E_P

Mesh	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$	$p = 128$	$p = 256$	$p = 512$
1	1.00	0.86	0.67	0.48	0.36	0.25	0.07	N/A	N/A	N/A
2	1.00	0.99	0.94	0.82	0.69	0.60	0.34	0.15	N/A	N/A
3	1.00	1.01	0.96	0.81	0.64	0.67	0.60	0.46	0.27	N/A
4	ET	ET	1.00	0.86	0.67	0.65	0.62	0.58	0.49	0.34



(a) Observed speedup S_p

(b) Observed efficiency E_p

Figure 3.1: Performance study of the CICR problem solved with first order FEM on maya 2013 by number of processes. Data based on 16 processes per node, except for the cases $p = 1, 2, 4, 8$. For the $128 \times 128 \times 512$ mesh, we use the modified definition $S_p = 4T_4(N)/T_p(N)$.

4 Performance Studies on maya 2010

This section describes the parallel performance studies for the solution of the CICR problem on the 2010 portion of maya. This portion consists of 84 nodes with two quad-core 2.8 GHz Intel Nehalem X5560 CPUs and 24 GB memory, connected by a dual-data rate (DDR) InfiniBand interconnect.

Table 4.1 collects the results of the performance study by number of nodes and processes per node. The table summarizes the observed wall clock time (total time to execute the code) in HH:MM:SS (hours:minutes:seconds) format. In situations where wall clock time are not obtained, ET indicates "excessive time required" (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$, where $N_z + 1$ is the number of finite volume cells on the z-direction for spatial mesh resolution of $N_x \times N_y \times N_z$.

Similar to Table 3.1, while reading along the first column of this sub-table, 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. We also observe significant improvement in runtime by doubling the number of processes from 4 to 8.

There are several differences between the results for maya 2013 and maya 2010. We use here the $64 \times 64 \times 256$ mesh in Tables 3.1 (c) and 4.1 (c) for discussion here, since these two sub-tables contain the most data. First, we compare wall clock times from 1 process per node to 8 processes per node. We notice the run times are faster on maya 1013 for all cases except for the case with 64 nodes and 4 processes per node. We also observe that the time difference for the same case is larger when using fewer number of nodes and processes per node. The better performance is due to faster CPUs on maya 2013. However, as we use more nodes and processes per node, the amount of work assigned to each process is reduced significantly. This is the reason why time differences between these two subtables on cases using 64 nodes are small. Now we look at results using 16 processes per node from Table 3.1 (c). With more CPU cores on the maya 2013 nodes, we can further improve the performance. For instance, using 8 nodes and 16 processes per node on maya 2013 we get significantly faster than 8 nodes and 8 processes per node, which in turn is already faster than results using 8 nodes on maya 2010.

Table 4.2 collects the results of the performance study by number of processes. Each row lists the results for one problem size. Each column corresponds to the number of parallel processes p used in the run. Data is based on 8 processes per node, except for the cases $p = 1, 2, 4$, where not all of the 8 cores of one node are utilized. This table demonstrates strong scalability on maya 2010. Table 4.2 (b) shows the speedup observed. The speedup S_p is increasing significantly as we increase the number of processes. We also observe that the speedup is better for larger problems. Table 4.2 (c) shows the observed efficiency E_p. Efficiency does not decrease significantly between $p = 8$ and $p = 16$ as we observed in Table 3.2 (c). Here we increased from 1 node to 2 nodes, where in the previous study we increased from 8 processes to 16 processes on the same node. The graphical representation of speedup and efficiency are presented in Figures 4.1 (a) and (b), respectively.

Table 4.1: Performance study of the CICR problem solved with first order FVM on maya 2010 by number of nodes and processes per node. ET indicates “excessive time required” (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$.

(a) Mesh resolution $N_x \times N_y \times N_z = 16 \times 16 \times 64$, DOF = 56,355							
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	00:13:29	00:07:03	00:03:44	00:02:25	00:01:22	00:01:01	00:00:55
2 processes per node	00:07:03	00:03:45	00:02:08	00:01:24	00:01:02	00:00:57	N/A
4 processes per node	00:03:41	00:02:06	00:01:19	00:01:00	00:00:49	N/A	N/A
8 processes per node	00:02:09	00:01:19	00:00:58	00:00:50	N/A	N/A	N/A

(b) Mesh resolution $N_x \times N_y \times N_z = 32 \times 32 \times 128$, DOF = 421,443							
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	02:57:08	01:28:17	00:44:00	00:25:36	00:12:25	00:07:02	00:04:37
2 processes per node	01:28:16	00:44:13	00:22:37	00:12:40	00:07:01	00:04:38	00:03:37
4 processes per node	00:47:20	00:23:27	00:12:17	00:06:52	00:04:29	00:03:26	N/A
8 processes per node	00:29:15	00:14:36	00:07:30	00:04:47	00:03:52	N/A	N/A

(c) Mesh resolution $N_x \times N_y \times N_z = 64 \times 64 \times 256$, DOF = 3,257,475							
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	30:51:54	15:40:07	07:53:15	03:54:49	01:59:23	01:04:22	00:35:20
2 processes per node	15:42:27	07:53:45	03:58:14	01:59:15	01:02:38	00:35:41	00:21:40
4 processes per node	08:52:53	04:28:33	02:10:39	01:20:37	00:40:29	00:22:58	00:15:54
8 processes per node	05:46:48	02:53:31	02:50:17	01:26:27	00:37:39	00:20:37	N/A

(d) Mesh resolution $N_x \times N_y \times N_z = 128 \times 128 \times 512$, DOF = 25,610,499							
	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	ET	ET	86:44:52	43:52:31	23:47:23	12:05:18	06:01:42
2 processes per node	ET	87:15:19	44:07:14	26:26:05	11:28:17	06:36:38	03:24:37
4 processes per node	94:34:57	48:05:38	32:11:15	12:43:22	08:01:20	04:07:11	02:14:17
8 processes per node	60:06:22	31:04:55	15:58:21	08:29:14	04:33:39	03:53:13	02:02:47

Table 4.2: Performance study of the CICR problem solved with first order FVM on maya 2010 by number of processes. Data based on 8 processes per node, except for the cases $p = 1, 2, 4$. Mesh 1 represents $16 \times 16 \times 64$, Mesh 2 represents $32 \times 32 \times 128$, Mesh 3 represents $64 \times 64 \times 256$, Mesh 4 represents $128 \times 128 \times 512$. ET indicates "excessive time required" (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$. For the $128 \times 128 \times 512$ mesh, we use the modified definition $S_p = 4T_4(N)/T_p(N)$.

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

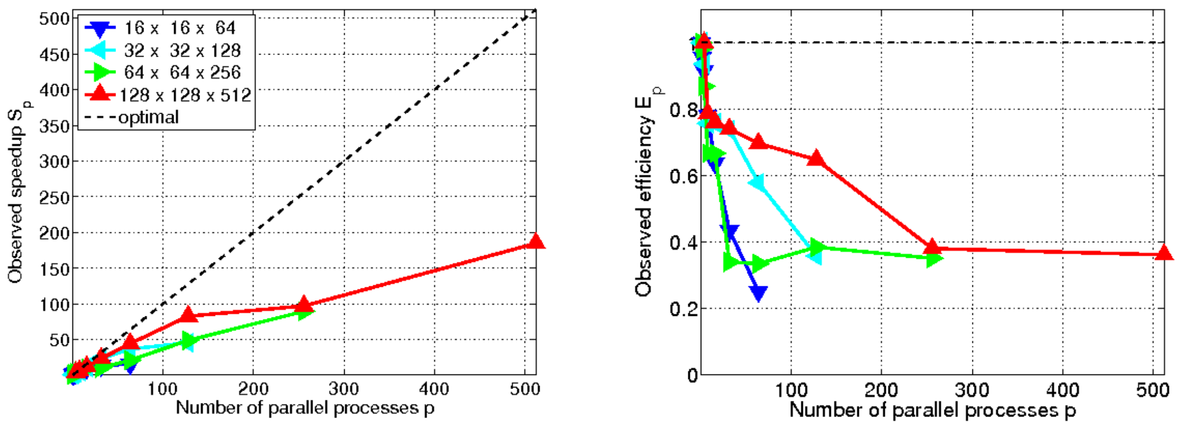
Mesh	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$	$p = 128$	$p = 256$	$p = 512$
1	00:13:29	00:07:03	00:03:41	00:02:09	00:01:19	00:00:58	00:00:50	N/A	N/A	N/A
2	02:57:08	01:28:16	00:47:20	00:29:15	00:14:36	00:07:30	00:04:47	00:03:52	N/A	N/A
3	30:51:53	15:42:27	08:52:53	05:46:48	02:53:31	02:50:17	01:26:27	00:37:39	00:20:37	N/A
4	ET	ET	94:34:57	60:06:22	31:04:55	15:58:21	08:29:14	04:33:39	03:53:13	02:02:47

(b) Observed speedup S_P

Mesh	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$	$p = 128$	$p = 256$	$p = 512$
1	1.00	1.91	3.67	6.26	10.20	13.90	16.06	N/A	N/A	N/A
2	1.00	2.01	3.74	6.06	12.14	23.64	37.00	45.80	N/A	N/A
3	1.00	1.96	3.48	5.34	10.67	10.88	21.42	49.18	89.83	N/A
4	ET	ET	4.00	6.29	12.17	23.69	44.58	82.95	97.33	184.88

(c) Observed efficiency E_P

Mesh	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$	$p = 128$	$p = 256$	$p = 512$
1	1.00	0.96	0.92	0.78	0.64	0.43	0.25	N/A	N/A	N/A
2	1.00	1.00	0.94	0.76	0.76	0.74	0.58	0.36	N/A	N/A
3	1.00	0.98	0.87	0.67	0.67	0.34	0.33	0.38	0.35	N/A
4	ET	ET	1.00	0.79	0.76	0.74	0.70	0.65	0.38	0.36



(a) Observed speedup S_p

(b) Observed efficiency E_p

Figure 4.1: Performance study of the CICR problem solved with first order FVM on maya 2010 by number of processes. Data based on 8 processes per node, except for the cases $p = 1, 2, 4$. For the $128 \times 128 \times 512$ mesh, we use the modified definition $S_p = 4T_4(N)/T_p(N)$.

5 Performance Studies on maya 2009

This section describes the parallel performance studies on maya 2009 for the solution of the CICR problem. This portion consists of 84 nodes with two quad-core 2.6 GHz Intel Nehalem X5550 CPUs and 24 GB memory, connected by a quad-data rate (QDR) InfiniBand interconnect.

Table 5.1 collects the results of the performance study by number of nodes and processes per node. The table summarizes the observed wall clock time (total time to execute the code) in HH:MM:SS (hours:minutes:seconds) format. In situations where wall clock time are not obtained, ET indicates "excessive time required" (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$, where $N_z + 1$ is the number of finite volume cells on the z-direction for spatial mesh resolution of $N_x \times N_y \times N_z$.

Similar to Table 4.1, while reading along the first column of this sub-table, 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. We also observe significant improvement in runtime by doubling the number of processes from 4 to 8.

There are some differences between the results for maya 2010 and maya 2009. We notice first the run times for the serial cases are slightly faster on maya 2010; this is expected since the CPUs on maya 2010 have a slightly faster clock rate than those on maya 2009. We also notice that while increasing the number of nodes, the performance of maya 2009 scales better. This difference is not surprising since maya 2009 is connected via a quad-data rate InfiniBand interconnect, while maya 2010 is connected via a dual-data rate InfiniBand.

Table 5.2 collects the results of the performance study by number of processes. Each row lists the results for one problem size. Each column corresponds to the number of parallel processes p used in the run. Data is based on 8 processes per node, except for the cases $p = 1, 2, 4$, where not all of the 8 cores of one node are utilized. This table demonstrates strong scalability on maya 2009. Table 5.2 (b) shows the speedup observed. The speedup S_p is increasing significantly as we increase the number of processes. We also observe that the speedup is better for larger problems. Table 5.2 (c) shows the observed efficiency E_p. Efficiency does not decrease significantly between $p = 8$ and $p = 16$ as we observed in Table 3.2 (c). Here we increased from 1 node to 2 nodes, where in the study with maya 2013 we increased from 8 processes to 16 processes on the same node. The graphical representation of speedup and efficiency are presented in Figures 5.1 (a) and (b), respectively.

Table 5.1: Performance study of the CICR problem solved with first order FVM on maya 2009 by number of nodes and processes per node. ET indicates “excessive time required” (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$.

(a) Mesh resolution $N_x \times N_y \times N_z = 16 \times 16 \times 64$, DOF = 56,355

	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	00:14:06	00:07:20	00:03:53	00:02:12	00:01:24	00:01:03	00:00:57
2 processes per node	00:07:17	00:03:54	00:02:13	00:01:25	00:01:03	00:01:00	N/A
4 processes per node	00:03:49	00:02:10	00:01:20	00:00:59	00:00:52	N/A	N/A
8 processes per node	00:02:13	00:01:23	00:00:58	00:00:48	N/A	N/A	N/A

(b) Mesh resolution $N_x \times N_y \times N_z = 32 \times 32 \times 128$, DOF = 421,443

	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	03:03:46	01:31:24	00:45:42	00:23:34	00:12:44	00:07:19	00:04:48
2 processes per node	01:31:32	00:45:49	00:23:31	00:12:42	00:07:19	00:04:53	00:03:47
4 processes per node	00:48:53	00:25:10	00:12:48	00:07:12	00:04:42	00:03:33	N/A
8 processes per node	00:30:14	00:14:59	00:07:53	00:04:53	00:03:49	N/A	N/A

(c) Mesh resolution $N_x \times N_y \times N_z = 64 \times 64 \times 256$, DOF = 3,257,475

	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	32:02:14	16:08:32	08:08:13	04:04:37	02:04:45	01:05:40	00:36:32
2 processes per node	16:12:44	08:10:32	04:03:09	02:04:31	01:05:12	00:36:19	00:22:32
4 processes per node	09:03:35	04:33:24	02:14:07	01:09:22	00:37:37	00:22:27	00:15:36
8 processes per node	05:49:56	02:58:11	01:26:44	00:45:40	00:26:36	00:17:57	N/A

(d) Mesh resolution $N_x \times N_y \times N_z = 128 \times 128 \times 512$, DOF = 25,610,499

	1 node	2 nodes	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1 process per node	ET	ET	88:20:34	44:57:55	23:02:04	11:54:24	06:08:25
2 processes per node	ET	88:54:29	44:52:09	23:01:33	11:51:04	06:08:52	03:23:21
4 processes per node	96:49:39	54:04:45	25:03:39	12:57:29	06:47:51	03:37:48	02:07:08
8 processes per node	61:13:18	31:30:29	16:16:17	08:35:04	04:38:24	02:38:24	01:46:25

Table 5.2: Performance study of the CICR problem solved with first order FVM on maya 2009 by number of processes. Data based on 8 processes per node, except for the cases $p = 1, 2, 4$. Mesh 1 represents $16 \times 16 \times 64$, Mesh 2 represents $32 \times 32 \times 128$, Mesh 3 represents $64 \times 64 \times 256$, Mesh 4 represents $128 \times 128 \times 512$. ET indicates "excessive time required" (more than 5 days), N/A indicates that the case is not feasible due to $p > (N_z + 1)$. For the $128 \times 128 \times 512$ mesh, we use the modified definition $S_p = 4T_4(N)/T_p(N)$.

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

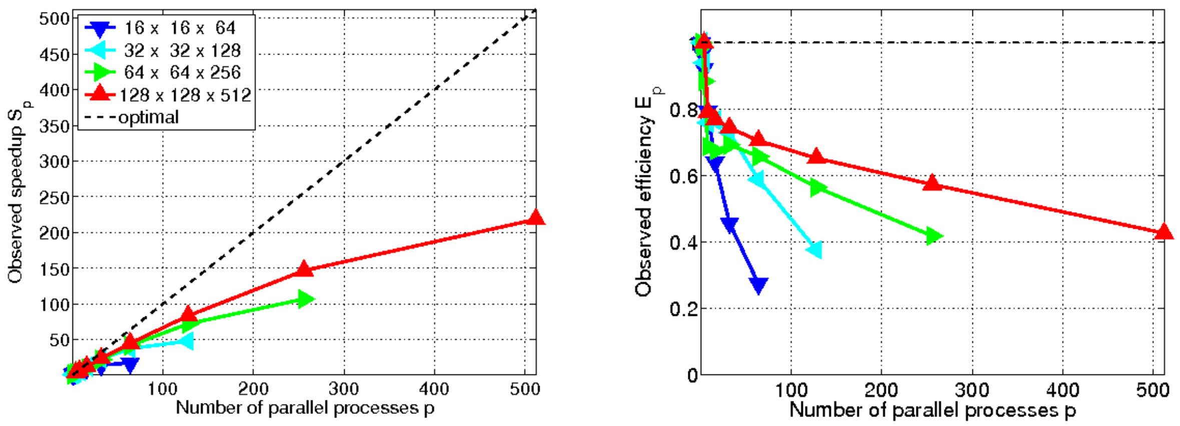
Mesh	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$	$p = 128$	$p = 256$	$p = 512$
1	00:14:06	00:07:17	00:03:49	00:02:13	00:01:22	00:00:58	00:00:48	N/A	N/A	N/A
2	03:03:46	01:31:32	00:48:53	00:30:14	00:14:59	00:07:53	00:04:53	00:03:49	N/A	N/A
3	32:02:14	16:12:44	09:03:35	05:49:56	02:58:11	01:26:44	00:45:40	00:26:36	00:17:57	N/A
4	ET	ET	96:49:39	61:13:18	31:30:29	16:16:17	08:35:04	04:38:24	02:38:23	01:46:25

(b) Observed speedup S_P

Mesh	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$	$p = 128$	$p = 256$	$p = 512$
1	1.00	1.94	3.69	6.35	10.26	14.65	17.51	N/A	N/A	N/A
2	1.00	2.01	3.76	6.08	12.27	23.33	37.63	48.20	N/A	N/A
3	1.00	1.98	3.54	5.49	10.79	22.16	42.09	72.27	107.04	N/A
4	ET	ET	4.00	6.33	12.29	23.80	45.12	83.47	146.72	218.39

(c) Observed efficiency E_P

Mesh	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$	$p = 128$	$p = 256$	$p = 512$
1	1.00	0.97	0.92	0.79	0.64	0.46	0.27	N/A	N/A	N/A
2	1.00	1.00	0.94	0.76	0.77	0.73	0.59	0.38	N/A	N/A
3	1.00	0.99	0.88	0.69	0.67	0.69	0.66	0.56	0.42	N/A
4	ET	ET	1.00	0.79	0.77	0.74	0.70	0.65	0.57	0.43



(a) Observed speedup S_p

(b) Observed efficiency E_p

Figure 5.1: Performance study of the CICR problem solved with first order FVM on maya 2009 by number of processes. Data based on 8 processes per node, except for the cases $p = 1, 2, 4$. For the $128 \times 128 \times 512$ mesh, we use the modified definition $S_p = 4T_4(N)/T_p(N)$.

6 Summary

Table 6.1 contains a summary of the results obtained on the cluster maya as well as a comparison to the previous HPCF cluster tara. The table reports results for the mesh resolution 64 × 64 × 256, since that is the largest mesh that could be solved on tara. The first three rows of the table contains results for the cluster tara, which was an 86-node distributed-memory cluster with two quad-core Intel Nehalem processors and 24 GB per node memory, a QDR InfiniBand interconnect, and 160 TB central storage. This cluster is part of the cluster maya as maya 2009, and its QDR InfiniBand interconnect extends to the newest portion maya 2013. The fourth row of the table contains results for maya 2009. The fifth row of the table contains results for the DDR InfiniBand connected portion maya 2010, and the sixth row contains results for the QDR InfiniBand connected portion maya 2013.

On the cluster tara, the default Portland Group compiler and the MVAPICH2 MPI implementation were used. The first row shows results from [12], which are obtained by using finite element method with QMR as linear solver, with no advection term in the model. We observe a close to optimal speedup of approximately 7 by running on all 8 cores rather than on one. We also observe a factor of approximately 28 speedup by increasing the number of nodes from 1 to 32, each with one processes per node. Finally, by using all 8 cores on 32 nodes we observe a speedup of 123. The second and third row of results were obtained in the preparation of [9]. The second row shows results obtained by using finite element method with BiCGSTAB as linear solver. We observe similar speedup as the first row. The third row shows results obtained by using finite volume method with BiCGSTAB as linear solver. The wall clock times are faster than the second row.

The fourth row contains results for maya 2009 after the tara cluster become a part of maya. The performance of maya 2009 is significantly better than tara. There are multiple reasons behind the difference, of which the most important ones are updated BIOS, firmware, operation system, and compiler versions. The results obtained on maya 2009 are compiled with the default environment of maya, which includes the latest Intel compiler and Intel MPI, as opposed to on tara we used the Portland Group compiler and MVAPICH2. This comparison implies how important it is to keep software up to date.

On maya 2010 and maya 2013, the problem is solved by the finite volume method with BiCGSTAB as linear solver, the same as on maya 2009, as well as the same Intel C compiler and MPI implementation. On maya 2010, we observe a less than optimal speedup of around 5.3 by running on all 8 cores rather than on one. We also observe a factor of approximately 28 speedup by increasing the number of nodes from 1 to 32, each with one processes per node. Using all 8 cores on 32 nodes we observe a speedup of 89.8. On maya 2013, we observe a speedup of around 10.3 by running on all 16 cores compare to on one. We also observe a factor of approximately 28 speedup by increasing the number of nodes from 1 to 32, each with one processes per node. On 32 nodes with 16 processes per node, $p = 512$ is larger than N_z and hence cannot be solved with this many processes; the table shows the result for 8 processes per node.

Looking at the first column of runtimes, the wall clock time for serial runs reduced significantly by improving numerical methods and using modern hardware. We also observe a factor of approximately 28 speedup by increasing the number of nodes from 1 to 32, consistently for all methods and all hardware. However, the second and last columns of runtimes indicate the speedup within each node decreased slightly, compared to the respective optimal values.

Table 6.1: Runtimes (speedup) for parabolic non-linear three-species CICR problem on the clusters tara and maya. We report results on the $N_x \times N_y \times N_z = 64 \times 64 \times 256$ mesh, since that is the largest mesh that could be solved on the previous cluster. To accommodate the requirement $p \leq (N_z + 1)$, the last column shows results with 8 processes per node also for maya (2013).

Cluster	method	serial	(1 node)	32 node	32 node
		1 core	all cores	1 core per node	all cores
		time	time (speedup)	time (speedup)	time (speedup)
tara (2009) [12]	FEM	57:14:46	08:19:29 (6.88)	01:59:36 (28.72)	00:27:47 (123.63)
tara (2009) [9]	FEM	67:04:28	09:17:32 (7.22)	02:15:03 (29.80)	00:29:06 (138.29)
tara (2009) [9]	FVM	47:46:46	07:31:51 (6.34)	01:41:46 (28.17)	00:25:54 (110.68)
maya (2009)	FVM	32:02:14	05:49:56 (5.49)	01:05:40 (29.27)	00:17:57 (107.09)
maya (2010)	FVM	30:51:54	05:46:48 (5.34)	01:04:22 (28.77)	00:20:37 (89.83)
maya (2013)	FVM	25:02:01	02:25:55 (10.29)	00:53:17 (28.19)	00:18:33 (80.97)

References

[1] Matthias K. Gobbert. Long-time simulations on high resolution meshes to model calcium waves in a heart cell. *SIAM J. Sci. Comput.*, vol. 30, no. 6, pp. 2922–2947, 2008.

[2] Jonathan Graf and Matthias K. Gobbert. Parallel performance studies for a parabolic test problem on the cluster maya. Technical Report HPCF–2015–7, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2015.

[3] Alexander L. Hanhart, Matthias K. Gobbert, and Leighton T. Izu. A memory-efficient finite element method for systems of reaction-diffusion equations with non-smooth forcing. *J. Comput. Appl. Math.*, vol. 169, no. 2, pp. 431–458, 2004.

[4] Xuan Huang and Matthias K. Gobbert. Parallel performance studies for a three-species application problem on maya 2013. Technical Report HPCF–2014–8, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2014.

[5] Leighton T. Izu, Joseph R. H. Mauban, C. William Balke, and W. Gil Wier. Large currents generate cardiac Ca^{2+} sparks. *Biophys. J.*, vol. 80, pp. 88–102, 2001.

[6] Leighton T. Izu, Shawn A. Means, John N. Shadid, Ye Chen-Izu, and C. William Balke. Interplay of ryanodine receptor distribution and calcium dynamics. *Biophys. J.*, vol. 91, pp. 95–112, 2006.

[7] Leighton T. Izu, W. Gil Wier, and C. William Balke. Evolution of cardiac calcium waves from stochastic calcium sparks. *Biophys. J.*, vol. 80, pp. 103–120, 2001.

[8] Samuel Khuvis and Matthias K. Gobbert. Parallel performance studies for an elliptic test problem on the cluster maya. Technical Report HPCF–2015–6, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2015.

[9] Jonas Schäfer, Xuan Huang, Stefan Kopecz, Philipp Birken, Matthias K. Gobbert, and Andreas Meister. A memory-efficient finite volume method for advection-diffusion-reaction systems with non-smooth sources. *Numer. Methods Partial Differential Equations*, vol. 31, no. 1, pp. 143–167, 2015.

[10] Thomas I. Seidman, Matthias K. Gobbert, David W. Trott, and Martin Kružík. Finite element approximation for time-dependent diffusion with measure-valued source. *Numer. Math.*, vol. 122, no. 4, pp. 709–723, 2012.

[11] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. *SIAM J. Sci. Comput.*, vol. 18, no. 1, pp. 1–22, 1997.

[12] David W. Trott and Matthias K. Gobbert. Parallel performance studies for a three-species application problem on the cluster tara. Technical Report HPCF–2010–11, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2010.