Received October 7, 2013, accepted November 5, 2013, date of publication November 12, 2013, date of current version November 21, 2013.

Digital Object Identifier 10.1109/ACCESS.2013.2290623

Solving Large Nonlinear Systems of First-Order Ordinary Differential Equations With Hierarchical Structure Using Multi-GPGPUs and an Adaptive Runge Kutta ODE Solver

AHMAD AL-OMARI[1], JONATHAN ARNOLD[3], THIAB TAHA[4], AND HEINZ-BERND SCHÜTTLER[2]
[1]Institute of Bioinformatics, University of Georgia, Athens, GA 30602, USA
[2]Department of Physics and Astronomy, University of Georgia, Athens, GA 30602, USA
[3]Department of Genetics, University of Georgia, Athens, GA 30602, USA
[4]Department of Computer Science, University of Georgia, Athens, GA 30602, USA

Corresponding author: A. Al-Omari (aomari@uga.edu)

This work was supported by the grants NSF QSB-0425762 and NSF DBI-1062213 and the Department of Systems Engineering and Medical Bioinformatics, Yarmouk University, Irbid, Jordan.

ABSTRACT The adaptive Runge–Kutta (ARK) method on multi-general-purpose graphical processing units (GPUs) is used for solving large nonlinear systems of first-order ordinary differential equations (ODEs) with over ∼10 000 variables describing a large genetic network in systems biology for the biological clock. To carry out the computation of the trajectory of the system, a hierarchical structure of the ODEs is exploited, and an ARK solver is implemented in compute unified device architecture/C++ (CUDA/C++) on GPUs. The result is a 75-fold speedup for calculations of 2436 independent modules within the genetic network describing clock function relative to a comparable CPU architecture. These 2436 modules span one-quarter of the entire genome of a model fungal system, *Neurospora crassa*. The power of a GPU can in principle be harnessed by using warp-level parallelism, instruction level parallelism or both of them. Since the ARK ODE solver is entirely sequential, we propose a new parallel processing algorithm using warp-level parallelism for solving ∼10 000 ODEs that belong to a large genetic network describing clock genome-level dynamics. A video is attached illustrating the general idea of the method on GPUs that can be used to provide new insights into the biological clock through single cell measurements on the clock.

INDEX TERMS Bioinformatics, biological clock, general-purpose graphical processing unit, finite element method, ordinary differential equation, adaptive Runge–Kutta integration, systems biology, warp-level parallelism.

I. INTRODUCTION

In a systems biology approach bridging genomics, bioinformatics, and engineering our goal is to explain the behavior of traits controlled by many genes, such as carbon metabolism, the biological clock, development, and cancer in terms of biochemical pathways found within living cells [1]. Since the 1990s, a variety of teams have assembled large maps of biochemical pathways in a variety of organisms with this goal in mind [2-4]. At the turn of the millennium it became possible to measure the dynamics of genomic-scale pathways spanning a whole living system [1, 5, 6]. We are now poised to describe the dynamics of an entire cell [7, 8]. A video is attached describing how this can be achieved through the integration of genomics, bioinformatics, and engineering [9].

Genetic networks describe time-dependent concentrations of molecular species, such as genes, their RNAs, and their proteins as well as their substrates [10]. These networks can be expressed as a system of coupled nonlinear first-order ordinary differential equations (ODEs). Understanding such networks enables us to discover the biochemistry and genetic activity of a cell and how the cell evolves as a function of time (including its metabolism, signal transduction, and cell cycle). Many problems could be solved and understood once these ODEs are identified. For example, human diseases like prostate cancer, the phenotype of other complex traits such as development [11], and the biological clock of an organism [12] could also be described. The most widely used approach to modeling these biochemical pathways are nonlinear systems of first-order ordinary differential equations [13].

GPUs have been used recently for solving computationally-intensive problems for many applications [14-17] including those in Bioinformatics [18, 19], numerical computations [20, 21], ray tracing [22], volume ray casting [23], computational fluid dynamics [24], and weather modeling [25]. Here we harness this new computing approach to develop new ODE solver methods employing Adaptive Runge Kutta Method (ARK) [26] on GPUs to simulate large genetic networks and ultimately identify these networks from available genomics data [12, 27-29].

A major proving ground for the new tools of systems biology has been the study of the molecular basis of the biological clock [30]. The key problem is linking the model identification of the clock to guiding expensive genomics experiments designed to identify the underlying network [13]. This model-guided discovery process, which we call computing life [12], requires the ability to simulate large nonlinear systems of first-order ODEs.

There are particular challenges to solving these ODEs. The system of ODEs is usually large. The experimental data are noisy and limited from molecular quantitative studies. More importantly, designing a new experiment is very expensive in terms of money (using genomics experiments) and time. To overcome the problem of many parameters and limited noisy data, new methods were developed for fitting these ODEs called ensemble methods [13, 27, 28]. The ensemble approach overcomes the limited genomics data on a particular network with many parameters by giving up on finding one best model. Instead, the search in the ensemble approach is for an ensemble of 40,000+ models consistent with the data. Averaging is then done over the ensemble to make predictions about the time-dependent behavior of the system. In order to implement these ensemble methods the ODE solver must be very fast!

Using the ARK method in the ensemble approach implies that the system of ODEs should be re-solved for each proposed ensemble Monte Carlo updating step, and solving for the time step t+h requires the solution at the prior t. For example, solving a genetic network as the one shown in (Fig. 1) [28] for the clock and constructing the ensemble of the unknown parameters that fit the experimental data needs a very large amount of time (i.e., 30 days on older processors). The diagram in (Fig. 2) specifies a much larger system of ODEs with hierarchical structure. There is a master clock module controlling 2436 slave modules each with 4 variables representing molecular species concentrations. We need a new approach to solve problems on this genomic scale.

Mainly, besides making these ensemble methods broadly available, our goal is to solve a genetic network shown in (Fig. 2) that consists of a master module (clock) and 2436 slave modules (subunits). Solving such a genetic network using a CPU implies that all of these subunits should be solved simultaneously and each subunit, solved many times sequentially. This makes the process of finding the unknown parameters in the network using the ensemble method massively time consuming. In some cases where the network consists of 2436 subunits [12], the ensemble method is beyond the capability of the fastest serial computers. We developed an algorithm using the concept of warp-level parallelism [31] with a GPU and ARK method that makes possible simulating 2436 subunits under clock control with a speed up of about 75-fold relative to a solution of serial version on a CPU architecture. The code (see supplement for code + input file) is written in C++/CUDA computer language for the GPU and is written in C++ and compiled with g++ using –O2 and –O3 optimization flags for the CPU. What makes our approach attractive is that as more subunits and ODEs are added, the speed up achieved increases, if we consider the availability of the GPUs. The strategy we describe here for solving large nonlinear systems of first-order ODEs is an alternative to another ODE solver recently developed [32].

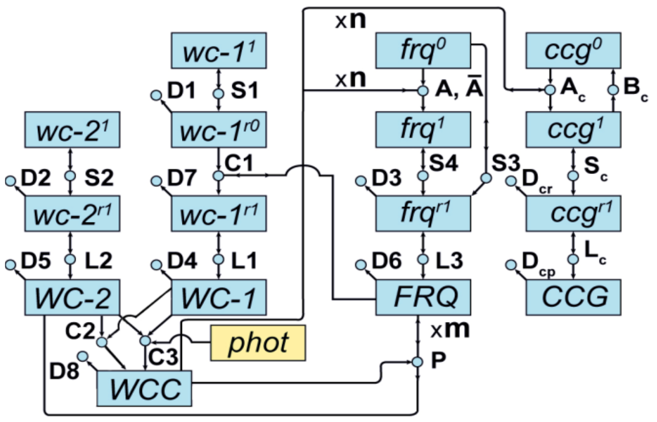


FIGURE 1. A genetic network for the biological clock from [28]. Molecular species (i.e., reactants or products) in the network are represented by boxes. The *white-collar-1* (*wc-1*), *white-collar-2* (*wc-2*), *frequency* (*frq*), and *clock controlled gene* (*ccg*) gene symbols are sometimes superscripted 0, 1, r0, r1, indicating, respectively, a transcriptionally inactive (0) or active (1) gene or a translationally inactive (r0) or active (r1) mRNA. Associated protein species are indicated with capitals. A phot (in yellow) symbolizes the photon species. Reactions in the network are represented by circles. Arrows pointing to circles identify reactants; arrows leaving circles identify products; and bi-directional arrows identify catalysts. The labels on each reaction, such as S4, also serve to denote the rate coefficients for each reaction. Reactions labeled with an S, L, or D denote transcription, translation, or degradation reactions, respectively. Reactions without products, such as D_7, are decay reactions. From [12].

II. METHODS

The Warp-level parallelism concept is used to exploit and harness the power of a GPU for solving 2436 systems of ODEs using the ARK method for a large genetic network shown in (Fig. 2) describing the biological clock in *N. crassa* [12]. Since such a genetic network consists of many subunits and all of these subunits have the same mathematical form (as ODEs) as shown below but with different parameters, solving all of these systems of ODEs once in parallel suits the SIMD

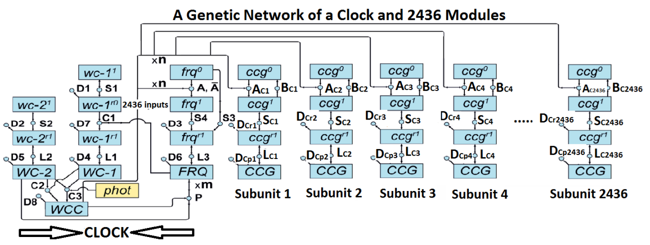


FIGURE 2. The whole genetic network consisting of 2436 slave modules (subunits) to be solved by the GPUs. The subunits are independent from each other but depend on the clock master module consisting of genes, *wc-1*, *wc-2*, *frq*, and their products. The subunits have the same mathematical form (ODEs) but different parameters. Identifying this huge genome scale network is beyond the fastest serial computer in the existence. Thus, the GPUs are necessary for solving such a network. This figure shows a modified genetic network for the biological clock from [28] in the genome. The notation to describe this network is the same as in (Fig. 1). An abbreviation of the notation for the *clock controlled genes* is now given: $g_0 = [ccg^0]$ = concentration of ccg^0; $g_1 = [ccg^1]$ = concentration of ccg^1; $g_r = [ccg^{r1}]$ = concentration of ccg^{r1}; g_p = [CCG] = concentration of CCG.

(single instruction, multiple data) and warp-level parallelism concepts (warp size for current NVIDIA GPUs is 32 threads). A common parallelization strategy in this category is to increase the number of warps and consequently the number of thread blocks (TBs) per streaming multiprocessor (SMX) on a GPU and decrease a TBs size (number of threads per block). In addition to the fact that this optimization strategy increases the number of thread blocks assigned to each SM, it provides more independent warps from other thread blocks when one warp is stalled [33]. (Fig. 2) shows 2436 systems of nonlinear ODEs (slave modules) that are needed to be solved to enable the implementation of the ensemble method with an ARK ODE solver [28]. The independence of these slave modules enabled us to suggest an algorithm to solve all of these modules in a parallel fashion using the ARK method and multi-GPGPUs.

In the Warp-level parallelism GPU(s) execute many warps concurrently. For example, on the Kepler K20x GPU, the maximum number of warps per SM equals to 64 warps, and the maximum number of TBs per SM equals to 16 TBs. Increasing the number of TBs and decreasing the block size is a well considered optimization strategy especially when the instruction level parallelism, i.e., thread code consists of multiple independent instructions in sequence, is hard to implement in some algorithms [33]. For example, the ARK method is in essence a sequential algorithm, and it is very hard to be parallelized by instructional level parallelism because ARK doesn't have independent instructions in sequence and because the time taken by each warp is unpredictable [20]. To maximize the usage of warp level parallelism we use a warp per block to solve the dynamics of the slave module consisting of a systems of nonlinear ODEs using the ARK method. The pressure of using a large number of blocks to solve our genetic network (2436 blocks) leads us to use multi-GPUs to increase the speed up as is shown in (Fig. 3) and (Fig. 4).

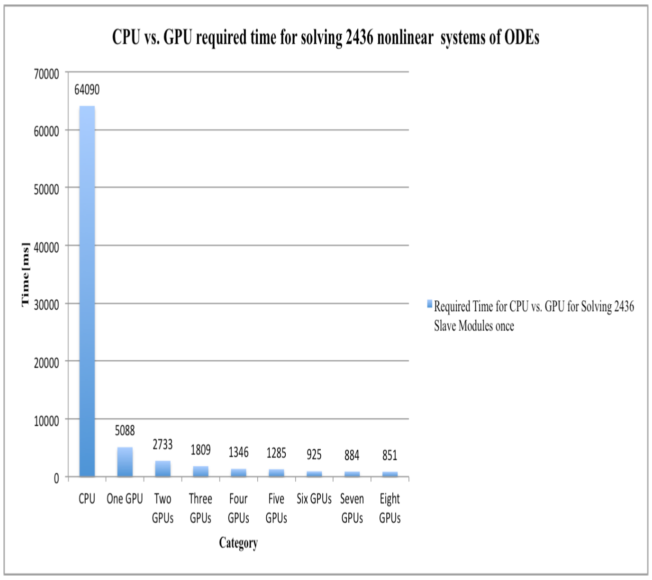


FIGURE 3. The time required for solving 2436 slave modules just one time using a NVIDIA GPU(s) [Kepler K-20x Tesla] over an extreme edition of optimized CPU [Quad-cores CPU [Intel(R) Core(TM) i5-2400 CPU@ 3.10GHz]. Using four of the GPUs to solve our target genetic network 800,000 times shown in the (Fig. 2) to fit the observed data requires just twelve days while using the CPU requires a year and six months.

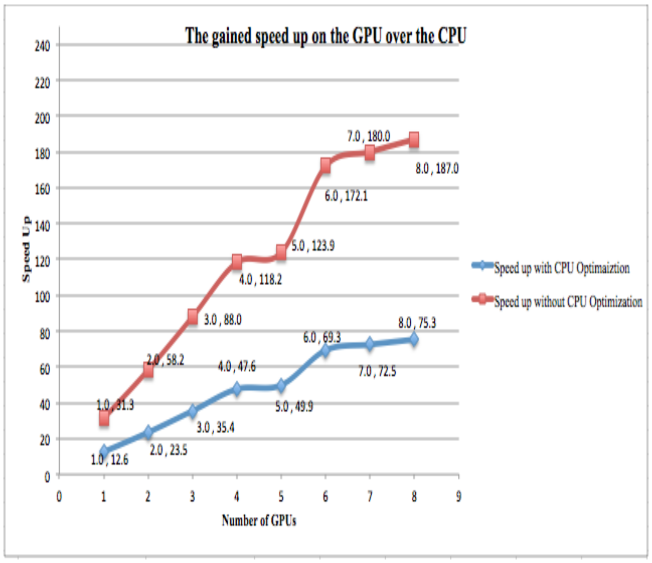


FIGURE 4. The achieved speed up using Multi-GPU [NVIDIA Kepler K-20x Tesla] over an extreme edition of CPU [Quad-cores CPU [Intel(R) Core(TM) i5-2400 CPU@ 3.10GHz]. Red line shows the speed up without using the –O2 and –O3 flags for CPU optimization (C++ code/g++ compiler) and blue line shows the speed up with using –O2 and –O3 flags for CPU optimization.

1) THE ALGORITHM

Each slave module can be described as an initial value problem of a system of nonlinear first ODEs for a genetic network as is shown in (Fig. 2) and is specified by

$$\frac{dg_0}{dt} = B_c g_1 - A_c g_0 w(t)$$
$$\frac{dg_1}{dt} = A_c g_0 w(t) - B_c g_1$$

$$\frac{dg_r}{dt} = S_c g_1 - D_{cr} g_r$$
$$\frac{dg_p}{dt} = L_c g_r - D_{cp} g_p$$

The variables in this subsystem are the concentrations of the *clock-controlled genes* (g_0 and g_1 in the inactive and active state, respectively), their mRNAs (g_r), and proteins (g_p). The testing was done on Quad-cores CPU [Intel(R) Core(TM) i5-2400 CPU@ 3.10GHz] Extreme Edition processor and a Tesla GPU [Kepler K-20x] to measure the speed up of our approach. A Kepler K-20x GPU handles double precision numbers and consists of 15 streaming multiprocessors (SMX), each (SMX) consisting of 192 SIMD cores and handling up to 16 TBs with restriction for 2048 threads per SMX. The idea of the algorithm is to assign each slave module to one TB consisting of 32 threads (a warp). The load of 2436 slaves modules (systems of nonlinear ODEs equations) are distributed equally across a Multi-GPUs system with a potential for a slight decrease or increase in the number of slave modules for the last GPU. For example, using four GPUs implies that to launch a kernel (a function to be executed on the GPU) on each GPU involves configurations of grid size equal to 609 thread blocks and a block size of 32 threads with a total number of threads equals to 19,488 threads per GPU. Each warp is responsible for solving a system of equations for one slave module. The number of slave modules (292) that are strongly supported to be under clock control was determined by a series of model-guided experiments [12].

2) THE PROCEDURE

1) Copying a file to the constant memory of each GPU that consists of a 200 time point solution to interpolate for the variable w(t) appearing in the equations above. Those time points come from the master clock module and are passed to each slave module as is shown in (Fig. 2). All of the TBs need to access the same file in the constant memory.
2) Each thread block executes a kernel, which contains the ARK ODE solver.
3) All threads in the same block hold the same data (*i.e.* initial conditions and parameters values), and execute the same system of equations for an assigned slave module.
4) ARK's constants are declared in the constant memory of the GPU and are seen by all of the threads.
5) Kernel configuration consists of a grid size equal to the number of slave modules (609 slave modules per GPU) and a block size equals to a warp size.

Parameter values of this clock network problem are given in Table1; all of the slave modules have the same set of parameters for the sake of simplicity and accuracy, and calculating performance of a CPU and GPU.

III. RESULTS

Identifying a large clock genetic network is beyond the capabilities of the fastest CPU ever manufactured and needs a much more expensive super computer than the GPUs that we use with high capabilities to solve such a network. The proposed parallel procedure uses NVIDIA Kepler K-20x GPU(s) to solve a genetic network shown in (Fig. 2) within a very short period of time compared with the time required on a CPU as is shown in (Fig. 3). The Ensemble method mentioned before needs this genetic network to be solved 40,000+ sweeps for an equilibration stage (each sweep is equivalent to solving the genetic network 10 times so that on average each variable out of ten variables in a slave module is updated once) and 40,000+ sweeps for an accumulations stage. The total number of sweeps to identify the genetic network of 2436 slave modules is shown in (Fig. 2) is 80,000+ sweeps. From Table 2, (Fig. 3), and (Fig. 4), the solution for 2436 slave modules from over 80,000 sweeps needs a CPU time of about one year and 6 months (considering 64090 milliseconds (ms) is needed to solve the 2436 slave modules once), while solving the same number of slave modules using just 4 GPUs needs about 12 days (considering 1346 ms is needed to solve the 2436 slave modules once), which is feasible and doable. Algorithm performance appears to plateau for 2436 slave modules somewhere between 4 and 6 GPUs in (Fig. 4). The genome dynamics of 295 *clock-controlled genes* over a 48 hour window are displayed in the attached video [9].

TABLE 1. Initial conditions at t=0 and parameters values.

Species	Initial Conditions	Parameters	Values
g_0	13.4271	Ac	0.3005
g_1	13.4348	Bc	37.2048
g_r	1.2208	Sc	0.0086
g_p	2.0982	Lc	11.4377
		Dcr	0.4105
		Dcp	0.3589

TABLE 2. A comparison of the simulation of a large clock genetic network on Kepler K-20x GPUs vs. a Quad-cores CPU [Intel(R) Core(TM) i5-2400 CPU@ 3.10GHz] Extreme Edition Processor with 2436 clock-controlled genes providing 2436 slave modules to the system of ODEs. The time required for solving the whole 2436 slave modules once using –O2 and –O3 optimization flags on the CPU equals to 64090 ms while without optimization flags equals to 159150 ms.

#of GPUs	GPU Time [ms]	Speed up With –O2, -O3 Flags	Speed up Without –O2, –O3 Flags
1	5088	12.59630503	31.27948113
2	2733	23.45042078	58.23271131
3	1809	35.42841349	87.97678275
4	1346	47.61515602	118.2392273
5	1285	49.87548638	123.8521401
6	925	69.28648649	172.0540541
7	884	72.5	180.0339367
8	851	75.31139835	187.0152761

A. ALGORITHM PERFORMANCE AS A FUNCTION OF THE NUMBER OF SLAVE MODULES IN THE CLOCK NETWORK

In previous work, we identified a total of 2436 genes that were circadian in the *N. crassa* genome [12], which is considerably more than the 292 reported *clock-controlled genes* [12]. The simplest null hypothesis to be investigated is that all 2436 genes that have a WCC binding site and are circadian in expression are in fact all *clock-controlled genes* [34]. To test this hypothesis with the ensemble method would involve being able to simulate the clock mechanism +2436 slave modules. We now do this and examine the computational time of the algorithm as a function of the number of slave modules up to 2436 such modules.

In (Fig. 5) we depict the relationship between different numbers of slave modules and time required for solving them using a fixed number of GPUs. Providing a brief introduction for the Kepler K-20x Tesla architecture and the programming model helps to elucidate the results in (Fig. 5) and (Fig. 6). The Kepler K-20x consists of 15 SMX with a maximum of TBs per streaming multiprocessor equals to 16 TBs. Thus, theoretically the device (GPU) can hold and run about 240 TBs simultaneously, given sufficient hardware resources, such as a register file and shared memory. Considering the sufficient resources, TBs do not leave a streaming multiprocessor until its execution is finished [35], and once TBs finish their time span, the scheduler keeps launching new TBs on these vacated SMXs for this kernel until all of the TBs have executed. For example, from (Fig. 5), on one hand, the time required on a single GPU for solving 200 or (800/4) slave modules or TBs (each slave module assigned to exactly a TB) equals to 462ms. On the other hand, the time required for solving 250 TBs (slave modules) needs 823ms, almost double the amount of time. This jump in the time is due to the fact that the device should solve theoretically 240 slave modules simultaneously until they finish their execution, and then it should start a new pass by solving the next 240 TBs or the rest of the available TBs simultaneously. Therefore, in this case, the 10 TBs difference need almost the same amount of time that is required for solving of 240 slave modules. As a matter of fact, although the theoretical number of TBs that can be run simultaneously on the device equals to 240, in our algorithm and as it is shown in (Fig. 5) and (Fig. 6), the number of blocks that can run simultaneously on the device is about 224 TBs, and after this number of blocks a jump in the time occurs for even one thread block more. The above clarifications should explain the scenario occurring on six GPUs as is shown in blue (Fig. 5) and (Fig. 6). Based on the fact that the time of the CPU is monotonically increasing with respect to the number of slave modules to be solved serially, then the drop in the speed up is shown in (Fig. 6) from time to time should be due to the jump in the time as is shown in (Fig. 5) given that the speed up=Time used by the GPU(s)/Time used by the CPU.

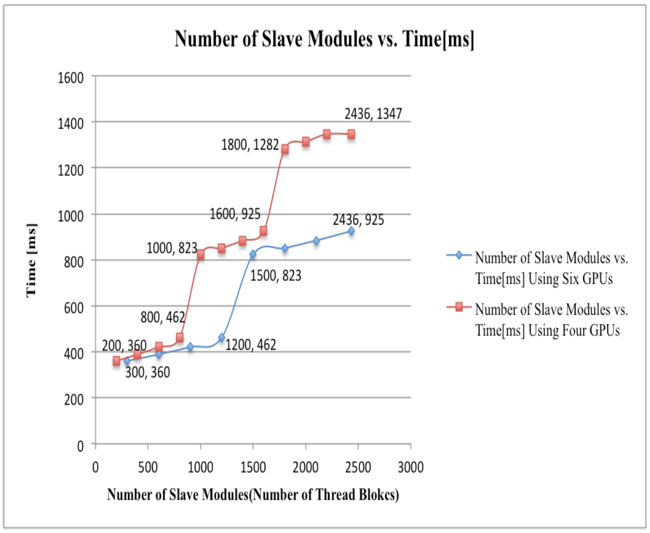


FIGURE 5. The relationship between number of thread blocks (slave modules) to be solved on the device and the required time. The jump in time is due to exceeding the maximum number of TBs running simultaneously on the device. If a GPU is capable of running N blocks simultaneously, then from 1 to N blocks takes the same time to complete.

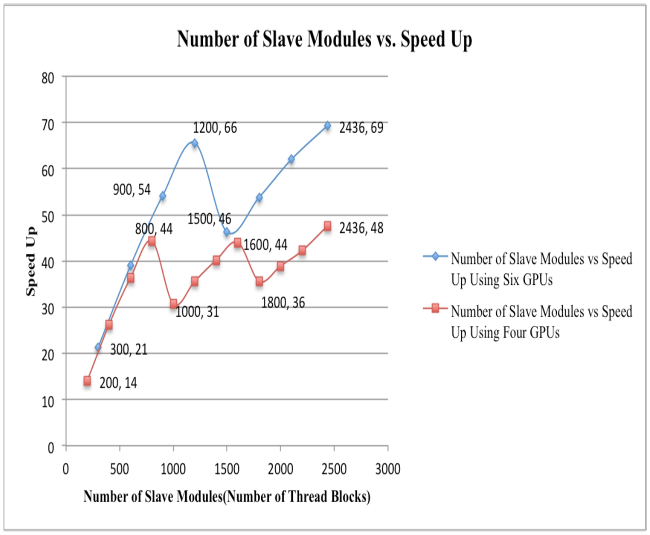


FIGURE 6. The relationship between number of thread blocks (slave modules) to be solved on the device and the speed up. The drop in the speed up is due to exceeding the maximum number of TBs running simultaneously on the device, given that the time of the CPU is monotonically increasing.

IV. DISCUSSION

This new parallelization strategy for solving large systems of ODEs on GPUs opens up the possibility of simulating genome dynamics. The parallelization strategy means that it is now feasible potentially to use ensemble methods for fitting genome-scale genetic networks when they have hierarchical structure. The need for such methods stems from genomics data being noisy and sparsely distributed across the genome [28]. The key to this parallelization strategy is to identify hierarchical structure in the genetic network. There is some experimental evidence that this hierarchical structure may be widespread [36], and has been argued to be a feature of the clock network [34]. This approach can be coupled with classic well-performing solvers, such as ARK, to make possible model-guided discovery about genetic networks on a genomic scale [12].

The GPU is used in [37] to solve a system of ODEs for another oscillatory system, and the maximum achieved speed up is 47-fold using a LSODA solver [38]. This system of ODEs consists of 3 species, 6 reactions, and 8 parameters. In this paper, our proposed parallel algorithm uses an ARK ODE solver to solve systems of ODEs and operates on a much larger genome scale, consisting of 9744 species, 14616 reactions, and 24360 parameters. Our GPU implementation also leads to a higher speed up reaching up to 75-fold.

Our approach can also be compared with an adaptive step size GPU ODE solver for simulating electric cardiac activity [39]. Garcia et al. [39] developed a method for solving systems of ∼300 ODEs describing cardiac activity with a single precision accuracy and a speed up of 9.07-fold while in our work here we solved a system of ∼10,000 ODEs using a double precision and a speed up of 75-fold.

An inherent limitation of the parallelization strategy here is the use of sequential ODE solvers, such as the ARK method. In order to calculate the trajectory at time t+h, it is necessary to have solved with high accuracy the trajectory at time t first. The ARK method is then inherently sequential. We recently developed an alternative parallelization strategy using a Galerkin Finite Element Method with Hat Functions as ODE solver [32], which is as accurate as the ARK method. It will be interesting to see how this alternative compares with parallelization using the hierarchical structure of the network. In comparison to [32], here in this paper another procedure using the ARK method along with the power of GPUs are functioning to solve the large genetic network using warp level parallelism, while in the Galerkin approach [32] is using instruction level parallelism alongside warp level parallelism in the GPUs. The only caveat is that the serial version of the Galerkin ODE solver is slower than ARK method and it is harder to implement on the GPUs.

A third parallelization strategy might involve a parallel implementation on multiple CPUs with MPI to narrow the gap in performance between CPUs and GPUs in Fig 4. We think the GPU is the preferred approach here for four reasons. One, using MPI across multiple CPUs needs hundreds of CPU cores, a more expensive strategy than multiple GPUs. Two, the thread on a GPU is very "lightweight" if it is compared with the thread on a CPU. Three, sometimes launching a certain number of threads on a CPU degrades the overall performance especially if the number of threads exceeds the number of cores. Finally, the best practice for implementation of ensemble methods has been serial implementations of solvers, and so we want to ascertain how the use of a GPU based ODE solver changes the speed of the ensemble method relative to best current practice.

This methodology of the ARK method on a GPU is enabling a new approach to understanding the kinetics of the cell on a genome scale. As captured in the attached video, new approaches in nanotechnology are enabling the measurement of the clock in single cells [9]. This will open a whole new area of inquiry about the clock. We can begin to ask if the clock is truly stochastic with variation in the oscillators from cell to cell and whether or not there is any cell-to-cell communication of oscillators in different cells. To address these questions will require the solution to three methodological challenges. New engineering approaches will be needed to make single cell measurements [40]. As indicated in the video, this approach involves capturing individual cells with microfluidics technology. New models will be needed to incorporate stochastic behavior in the clock models [41]. While stochastic clock models have been proposed, they have no empirical basis and have not been tested. Third, new parallelization strategies will be needed to understand the large networks describing clock behavior. The clock network in a single cell could involve potentially 2436 distinct genes responding to the clock mechanism or equivalently, 1/4 of the genome [12]. In this paper, we have introduced the second of two strategies to address the fitting of genome scale networks by the ensemble method. The time estimates in Table 1 implies that it is now feasible to fit a genome-scale network to genome dynamics within a single cell, like that of the clock, with ensemble methods needed to overcome noisy data that is sparsely distributed across the genome. A 75-fold speedup of the simulation of a hierarchical network on GPUs was achievable (Table 1). This speedup is sufficient to fit the entire clock network to the genome dynamics of a single cell.

V. CONCLUSION

In this paper we harness the power of multi-GPGPUs to solve many systems of nonlinear ordinary differential equations that belong to a large genetic network describing clock genome-level dynamics using an Adaptive Runge Kutta ODE solver. Implementing the proposed algorithm opens up a door to utilize the ensemble approach to overcome the problem of many parameters and limited noisy genomics data on a particular network. Consequently, understanding such networks enables us to discover the biochemistry and genetic activity of a cell and how the cell evolves as a function of time (including its metabolism, signal transduction, and cell cycle).

REFERENCES

[1] T. Ideker, V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, J. K. Eng, *et al.*, "Integrated genomic and proteomic analyses of a systematically perturbed metabolic network," *Science*, vol. 292, no. 5518, pp. 929–934, May 2001.

[2] I. M. Keseler, J. Collado-Vides, A. Santos-Zavaleta, M. Peralta-Gil, S. Gama-Castro, L. Muniz-Rascado, *et al.*, "EcoCyc: A comprehensive database of *Escherichia coli* biology," *Nucl. Acids Res.*, vol. 39, pp. D583–D590, Jan. 2011.

[3] S. Okuda, T. Yamada, M. Hamajima, M. Itoh, T. Katayama, P. Bork, *et al.*, "KEGG Atlas mapping for global analysis of metabolic pathways," *Nucl. Acids Res.*, vol. 36, pp. W423–W426, Jul. 2008.

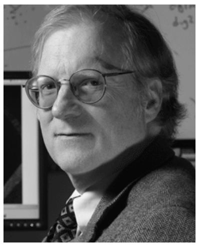
[4] R. Overbeek, N. Larsen, G. D. Pusch, M. D'Souza, E. Selkov, N. Kyrpides, *et al.*, "WIT: Integrated system for high-throughput genome sequence analysis and metabolic reconstruction," *Nucl. Acids Res.*, vol. 28, no. 1, pp. 123–125, Jan. 2000.

[5] J. L. DeRisi, V. R. Iyer, and P. O. Brown, "Exploring the metabolic and genetic control of gene expression on a genomic scale," *Science*, vol. 278, no. 5338, pp. 26–37, Oct. 1997.

[6] S. J. Maerkl and S. R. Quake, "A systems approach to measuring the binding energy landscapes of transcription factors," *Science*, vol. 315, no. 5809, pp. 233–237, Jan. 2007.

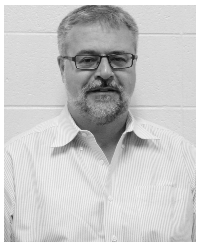
[7] M. W. Covert, E. M. Knight, J. L. Reed, M. J. Herrgard, and B. O. Palsson, "Integrating high-throughput and computational data elucidates bacterial networks," *Nature*, vol. 429, no. 6987, pp. 82–96, May 2004.
[8] J. R. Karr, J. C. Sanghvi, D. N. Macklin, M. V. Gutschow, J. M. Jacobs, B. Bolival, *et al.*, "A whole-cell computational model predicts phenotype from genotype," *Cell*, vol. 150, no. 2, pp. 389–401, Jul. 2012.
[9] T. R. Dickey, "Single cell measurements on the biological clock by microfluidics,"
[10] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, no. 6825, pp. 268–276, Mar. 2001.
[11] J. Jaeger, S. Surkova, M. Blagov, H. Janssens, D. Kosman, and K. N. Kozlov, "Dynamic control of positional information in the early *Drosophila* embryo," *Nature*, vol. 430, pp. 368–371, Jul. 2004.
[12] W. Dong, X. Tang, Y. Yu, R. Nilsen, R. Kim, J. Griffith, *et al.*, "Systems biology of the clock in *Neurospora crassa*," *PloS One*, vol. 3, no. 8, pp. e3105-1–e3105-3, Aug. 2008.
[13] A. K. Chakraborty and J. Das, "Pairing computation with experimentation: A powerful coupling for understanding T cell signalling," *Nature Rev. Immunol.*, vol. 10, pp. 59–71, Jan. 2010.
[14] D. Luebke, M. Harris, N. Govindaraju, A. Lefohn, M. Houston, J. Owens, *et al.*, "GPGPU: General-purpose computation on graphics hardware," in *Proc. ACM/IEEE Conf. Supercomput.*, Nov. 2006, p. 208.
[15] S. Ohshima, K. Kise, T. Katagiri, and T. Yuba, "Parallel processing of matrix multiplication in a CPU and GPU heterogeneous environment," in *High Performance Computing for Computational Science-VECPAR* (Lecture Notes in Computer Science). New York, NY, USA: Springer-Verlag, 2007, pp. 305–318.
[16] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, *et al.*, "A survey of general-purpose computation on graphics hardware," *Comput. Graph. Forum*, vol. 26, no. 1, pp. 80–113, 2007.
[17] C. J. Thompson, S. Hahn, and M. Oskin, "Using modern graphics architectures for general-purpose computing: A framework and analysis," in *Proc. 35th Annu. ACM/IEEE Int. Symp. Microarchit.*, Aug. 2002, pp. 306–317.
[18] W. Liu, B. Schmidt, and W. Müller-Wittig, "Performance analysis of general-purpose computation on commodity graphics hardware: A case study using bioinformatics," *J. VLSI Signal Process. Syst. Signal, Image, Video Technol.*, vol. 48, no. 3, pp. 209–221, Sep. 2007.
[19] M. Charalambous, P. Trancoso, and A. Stamatakis, "Initial experiences porting a bioinformatics application to a graphics processor," in *Advances in Informatics*. New York, NY, USA: Springer-Verlag, 2005, pp. 415–425.
[20] L. Murray, "GPU acceleration of Runge-Kutta integrators," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 94–101, Jan. 2012.
[21] G. Khanna, "High-precision numerical simulations on a CUDA GPU: Kerr black hole tails," *J. Sci. Comput.*, vol. 56, no. 2, pp. 1–15, Aug. 2013.
[22] C. Gribble and A. Naveros, "GPU ray tracing with rayforce," in *Proc. ACM SIGGRAPH*, Jul. 2013, p. 98.
[23] T. L. Falch, J. B. Floystad, D. W. Breiby, and A. C. Elster, "GPU-accelerated visualization of scattered point data," *IEEE Access*, vol. 1, no. 9, pp. 564–576, Sep. 2013.
[24] S. Park and H. Shin, "Efficient generation of adaptive Cartesian mesh for computational fluid dynamics using GPU," *Int. J. Numer. Methods Fluids*, vol. 70, no. 11, pp. 1393–1404, Dec. 2012.
[25] I. Demeshko, N. Maruyama, H. Tomita, and S. Matsuoka, "Multi-GPU implementation of the NICAM atmospheric model," in *Proc. Euro-Par Parallel Process. Workshops*, 2013, pp. 175–184.
[26] E. W. Cheney and D. R. Kincaid, *Numerical Mathematics and Computing*. Pacific Grove, CA, USA: Brooks/Cole, 2012.
[27] D. Battogtokh, D. K. Asch, M. E. Case, J. Arnold, and H. B. Schüttler, "An ensemble method for identifying regulatory circuits with special reference to the *qa* gene cluster of *Neurospora crassa*," *Proc. Nat. Acad. Sci. United States Amer.*, vol. 99, no. 26, pp. 16904–16909, Dec. 2002.
[28] Y. Yu, W. Dong, C. Altimus, X. Tang, J. Griffith, M. Morello, *et al.*, "A genetic network for the clock of *Neurospora crassa*," *Proc. Nat. Acad. Sci. United States Amer.*, vol. 104, no. 8, pp. 2809–2814, Feb. 2007.
[29] X. Tang, W. Dong, J. Griffith, R. Nilsen, A. Matthes, K. B. Cheng, *et al.*, "Systems biology of the *qa* gene cluster in *Neurospora crassa*," *PloS One*, vol. 6, no. 6, pp. e20671-1–e20671-3, Jun. 2011.
[30] H. Ukai and H. R. Ueda, "Systems biology of mammalian circadian clocks," *Annu. Rev. Physiol.*, vol. 72, pp. 579–603, Mar. 2010.
[31] J. Passerat-Palmbach, J. Caux, P. Siregar, and D. R. C. Hill, "Warp-level parallelism: Enabling multiple replications in parallel on GPU," in *Proc. Eur. Simul. Model. Conf.*, 2011, pp. 24–26.
[32] A. Al-Omari, H. B. Schuttler, J. Arnold, and T. Taha, "Solving nonlinear systems of first order ordinary differential equations using a galerkin finite element method," *IEEE Access*, vol. 1, no. 7, pp. 408–417, Jul. 2013.
[33] S. Ryoo, "Program optimization strategies for data-parallel many-core processors," Ph.D. dissertation, Dept. Electr. Comput. Eng., Univ. Illinois at Urbana-Champaign, Urbana, IL, USA, 2008.
[34] C.-H. Chen, C. S. Ringelberg, R. H. Gross, J. C. Dunlap, and J. J. Loros, "Genome-wide analysis of light-inducible responses reveals hierarchical light signalling in Neurospora," *EMBO J.*, vol. 28, no. 8, pp. 1029–1042, Apr. 2009.
[35] L. Chen, O. Villa, S. Krishnamoorthy, and G. R. Gao, "Dynamic load balancing on single-and multi-GPU systems," in *Proc. IEEE IPDPS*, Apr. 2010, pp. 1–12.
[36] D. Segre, A. DeLuna, G. M. Church, and R. Kishony, "Modular epistasis in yeast metabolism," *Nature Genet.*, vol. 37, no. 1, pp. 77–83, Jan. 2004.
[37] Y. Zhou, J. Liepe, X. Sheng, M. P. H. Stumpf, and C. Barnes, "GPU accelerated biochemical network simulation," *Bioinformatics*, vol. 27, no. 6, pp. 874–876, Mar. 2011.
[38] A. C. Hindmarsh, *ODEPACK—A Systematized Collection of ODE Solvers*, vol. 1, R. S. Stepleman, Ed. Amsterdam, The Netherlands: North Holland, 1983, pp. 55–64.
[39] V. M. Garcia, A. Liberos, A. M. Climent, A. Vidal, J. Millet, and A. Gonzalez, "An adaptive step size GPU ODE solver for simulating the electric cardiac activity," in *Proc. Comput. Cardiol. Conf.*, Sep. 2011, pp. 233–236.
[40] G. M. Whitesides, "The origins and the future of microfluidics," *Nature*, vol. 442, pp. 368–373, Jul. 2006.
[41] D. Gonze, J. Halloy, and A. Goldbeter, "Robustness of circadian rhythms with respect to molecular noise," *Proc. Nat. Acad. Sci. United States Amer.*, vol. 99, no. 2, pp. 673–678, Jan. 2002.

AHMAD AL-OMARI received the B.Sc. degree in electrical and computer engineering from Yarmouk University, Irbid, Jordan, in 2004. He was a Teaching Assistant and Lab Engineer with the Department of Electrical and Computer Engineering, Yarmouk University, from 2004 to 2010. He was involved in more than 30 different projects on computers, communication, and electronic engineering. He received a grant of over $100 000 from Yarmouk University to pursue the Ph.D. degree in bioinformatics. He is currently a Research and Teaching Assistant and the Ph.D. degree in bioinformatics under the supervision of Prof. J. Arnold with the University of Georgia, Atlanta, GA, USA, on systems biology and genetic networks. His current research interests include parallel computation, systems biology, finite elements method, machine learning and pattern recognition, biological circuits and gene networks, and numerical analysis.



JONATHAN ARNOLD received the B.S. degree in mathematics from Yale University in 1975 and the M.Phil. and Ph.D. degrees in statistics from Yale University in 1978 and 1982, respectively. In 1982, he joined the Departments of Statistics and Genetics, College of Arts and Sciences, University of Georgia, where he has risen through the ranks to professor of Genetics, Statistics, and Physics and Astronomy. He has authored or co-authored over 130 papers in journals, book chapters, and conference proceedings in the subjects of statistical genetics, population genetics, computational biology, and fungal genomics. He has served on the NSF Computational Biology Panel, DOE Genome Panel, and as a charter member on the NIH Genetic Variation and Evolution Study Section from 2005 to 2008. His research interests include the development and identification of genetic networks of fundamental processes in the model system *Neurospora crassa*, i.e., Computing Life. He was elected an AAAS fellow in 2011.

THIAB TAHA received the Ph.D. degree from Clarkson University in 1982. In 1982, he joined the Computer Science Department, University of Georgia. His research interests include scientific and distributed computing and software development for solving problems in nonlinear waves, optical fiber communication systems, biochemical reaction networks, and related topics. He has authored more than 60 research papers. He is a Senior Editor of the *Mathematics and Computers in Simulation Journal* and the Vice President of IMACS. He received the M. G. Michael Award for Research in the Sciences at UGA in 1985 and was a Fulbright scholar from 1995 to 1996. Currently, he is the Director of the CUDA Teaching Center, UGA.



HEINZ-BERND SCHÜTTLER received the Diplom degree in physics from Technische Universität München, Germany, in 1981, and the Ph.D. degree in physics from the University of California Los Angeles in 1984. After post-doctoral appointments at the University of California Santa Barbara and at the Argonne National Laboratory, he joined the faculty of the Department of Physics and Astronomy, University of Georgia, in 1987, where he currently serves as a Professor of physics. His research interests include applications of computational statistical mechanics in biology and condensed matter physics. His recent notable contributions in computational biology include the development of the ensemble network simulation (ENS) method for the reconstruction of biological circuits; and ENS-based methods for maximally informative next experiment design.