УДК 537.86, 530.182

МОДЕЛИРОВАНИЕ ДИНАМИКИ ИММУННОГО ОТВЕТА В АНСАМБЛЯХ Т-КЛЕТОК С ПРИМЕНЕНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

Д.Ю. Зорин, М.В. Иванченко

Нижегородский госуниверситет им. Н.И. Лобачевского

laonden@gmail.com

Поступила в редакцию 27.02.2013

С использованием технологий параллельных вычислений разработана программа, которая позволила исследовать в численном эксперименте динамику математической модели многоклональной сети Т-лимфоцитов и функцию иммунопротеасомы. Проведен анализ нескольких методов распараллеливания. Использование графического процессора позволило существенно уменьшить длительность вычислений по сравнению с аналогичной реализацией на центральном процессоре.

Ключевые слова: конкуренция, динамика клеточных сетей, математическая иммунология, параллельные вычисления, CUDA, OpenMP.

Введение

Современное численное моделирование сталкивается с проблемой быстродействия вычислительных блоков, тактовая частота которых ограничена. Кардинально увеличить производительность вычислений возможно только при увеличении количества ядер процессора при распараллеливании алгоритма вычислений. Если программа состоит из последовательных и параллельных участков кода, то максимальная масштабируемость производительности будет ограничена долей времени, которое уходит на обработку последовательного участка кода. Если же таких участков мало, то в принципе производительность будет увеличиваться пропорционально количеству ядер многоядерного процессора. Поскольку современные видеокарты способны объединять сотни ядер, в отличие от центральных процессоров, использование в качестве вычислителя графического процессора (англ. *GPU, graphics processing units*) дает огромные преимущества, которые могут быть использованы для решения целого ряда вычислительно сложных задач. К таким задачам относятся матричные операции [1], задачи обработки изображений [2], преобразования Фурье [3], машинного зрения и т.д. [4].

Технология использования графического процессора для общих вычислений (англ. *GPGPU, general-purpose graphics processing units*) [5] на сегодняшний день имеет несколько популярных реализаций:

- CUDA [6] – технология, позволяющая реализовывать алгоритмы, выполняемые на графических процессорах. Технология CUDA разработана компанией Nvidia.
- OpenCL [7] – язык программирования задач, связанных с параллельными вычислениями на различных графических и центральных процессорах.

Широко используемой технологией распараллеливания на CPU является технология OpenMP [8], которая предназначена для программирования многопоточных приложений на многопроцессорных системах с общей памятью. За основу берется последовательная программа, а для создания ее параллельной версии пользователю предоставляется набор директив, функций и переменных окружения. OpenMP реализует параллельные вычисления с помощью многопоточности, в которой «главный» (master) поток создает набор «подчиненных» (slave) потоков, и задача распределяется между ними [9]. OpenMP максимально подходит для эффективной реализации параллельных программ современных многоядерных процессоров [10].

Одним из направлений использования технологий распараллеливания является математическое моделирование динамики клеточных ансамблей, как в теоретических, так и в прикладных биомедицинских задачах, в том числе в иммунологии. Это связано с тем, что постановка прямых экспериментов *in vitro* и *in vivo* в этой области зачастую крайне сложна, если вообще возможна. В таком случае применяются эксперименты *in silico* [11].

Иммунная система борется с патогенами посредством Т-лимфоцитов (Т-клетки), которые имеют на своей поверхности специальные рецепторы (англ. *TCR, T-cell receptor*). Благодаря этим рецепторам Т-клетки специфичны к конкретным антигенам и постоянно контактируют

с макрофагами, В-клетками, дендритными клетками, которые несут на молекулах главного комплекса гистосовместимости (англ. *MHC, major histocompatibility complex*) антигены. Т-клетки имеют антиген-специфические свойства, т.е. определенные Т-клетки могут узнавать определенные пептиды (участки других молекул – антигенов).

Т-клетки в отсутствие инфекции находятся в состоянии гомеостаза [12]. Число Т-лимфоцитов у здорового взрослого человека оценивается как 10^{10}–10^{11}, а их вариантов TCR (клонотипов) 10^{7}–10^{8} [13]. Т-клетки обладают перекрестной реактивностью, т.е. отдельный клонотип может распознавать сразу несколько пептидов [14] и, следовательно, конкурировать с другими клонотипами за стимулы – слабые сигналы, получаемые от взаимодействия TCR с распознаваемыми MHC. Если Т-клетка получает такой сигнал, то она делится. Принято разделять Т-клетки на наивные, еще не принимавшие участия в процессе распознавания чужеродных антигенов, и на эффекторные, которые участвуют в ликвидации чужеродного материала (активируются из наивных Т-лимфоцитов).

Основным инструментом процессинга пептидов является протеасома – белковый комплекс, основной компонент убиквитин-зависимой деградации ненужных и повреждённых белков [15]. Существует несколько видов протеасом, среди которых: конституционная, доминирующая в отсутствие инфекции (во всех клетках организма, кроме дендритных клеток), и иммунная, получаемая из конституционной, например, после стимуляции γ-интерфероном [16], который выделяется клетками в ответ на присутствие различных патогенов.

Экспериментально установлено, что иммунопротеасома «нарезает» белки на пептиды несколько иначе, чем конституционная, однако роль ее до сих пор неизвестна. По одной из гипотез за счет работы иммунопротеасомы увеличивается действенность иммунного ответа [17].

В предыдущей работе [18] мы рассмотрели математическую модель, описывающую функцию протеасомы в соответствии с этой гипотезой, и сделали предположение о степенном законе распределения пептидных концентраций для большого количества клонов. Но для статистической достоверности распределения требуется большая матрица пептидных связей (порядка $(10^{3}–10^{4})\times(10^{3}–10^{4})$), что, в свою очередь, обусловливает крайне высокую трудоемкость вычислений, следовательно, появляется необходимость в использовании технологий параллельных вычислений.

Цель данной работы – программная реализация математической модели иммунопротеасомной «нарезки» белков на пептиды с использованием технологий CUDA и OpenMP, сравнение их производительности как для высокопроизводительного кластера, так и для персонального компьютера.

Математическая модель

На основе изложенных выше представлений можно сформулировать математическую модель, которая описывает конкурентную динамику наивных и эффекторных клеток в отсутствие и в присутствии инфекции и представляет собой систему обыкновенных дифференциальных уравнений типа молекулярной кинетики [17, 19]:

$$\begin{cases} \dfrac{dN_i}{dt} = H_i + (2\varphi - 1)\rho_N \sum\limits_{j=1}^{N_{pep}} K_{ij} F_j N_i - \delta_N N_i, \\ \dfrac{dE_i}{dt} = 2(1-\varphi)\rho_N \sum\limits_{j=1}^{N_{pep}} K_{ij} F_j N_i + \rho_E \sum\limits_{j=1}^{N_{pep}} K_{ij} F_j E_i - \delta_E E_i, \\ F_j = \dfrac{M_T P_j}{1 + \sum_{i=1}^{N_{clon}} K_{ij}(N_i + E_i)}, \end{cases}$$

где N_i – количество клеток в *i*-м клонотипе пула наивных клеток; E_i – количество клеток в *i*-м клонотипе пула эффекторных клеток; F – количество свободного места на антигенпрезентующей клетке (АПК); P_j – пептидная концентрация (доля *j*-го пептида); K – матрица силы связей *i*-го клонотипа с *j*-м пептидом (часть пептидов может быть патогенна, а остальная – нейтральна), размер матрицы K – $N_{clon}\times N_{pep}$, где N_{clon} – количество клонотипов и N_{pep} – количество пептидов в системе; δ_N, δ_E – скорость вымирания наивных и эффекторных клеток соответственно; ρ_N, ρ_E – коэффициенты пролиферации; φ – доля обновления наивных клеток.

Первое уравнение описывает динамику наивных клеток, часть из которых (2–2φ) переходит в пул эффекторных. Также уравнения взаимосвязаны через количество свободного места на АПК. Когда количество Т-клеток, которые распознают *j*-й пептид, возрастает, больше MHC становится занято Т-клетками и количество свободного места сокращается.

Чужеродный патогенный пептид представлен в модели только тогда, когда появляется инфекция и, соответственно, инфицированные клетки. При больших количествах инфицированных клеток доля патогенного пептида P_v достигает максимальной величины P_0: $P_v(t) = \dfrac{P_0 I(t)}{I(t)+\theta}$, здесь θ – чувствительность презентации патогенов. Мы предполагаем наличие только одного патогенного пептида.

Таблица 1

Параметр	Описание	Размерность	Значение
δ_N	Скорость вымирания наивных клеток	день$^{-1}$	0.005
δ_E	Скорость вымирания эффекторных клеток	день$^{-1}$	0.4
φ	Доля обновления наивных клеток		0.99
k_h	Сила распознавания чужих пептидов	день$^{-1}$ место$^{-1}$	5×10^{-2}
k_s	Сила распознавания своих пептидов	день$^{-1}$ место$^{-1}$	5×10^{-5}
ρ_E	Скорость пролиферации эффекторных клеток	день$^{-1}$	100
ρ_N	Скорость пролиферации наивных клеток	день$^{-1}$	10
H_i	Скорость выхода клеток из тимуса	день$^{-1}$ клон$^{-1}$	1
M_T	Количество «места» на главном комплексе гистосовместимости		10^5
θ	Чувствительность презентации патогенов		1
P_0	Максимальная возможная доля патогена		0.07
r	Репликация патогена	день$^{-1}$	10
c	Объем патогена	клетка$^{-1}$	10^{-7}
D	Очистка инфицированных клеток	день$^{-1}$ клетка$^{-1}$	5×10^{-5}

Динамика количества инфицированных клеток I описана стандартной моделью [17, 19]: $\frac{dI}{dt} = r(1-cI)I - D(\sum_{i=1}^{N_{clon}} K_{iP}E_i)I$, где r – коэффициент репликации патогена; c – объем патогена; D – скорость удаления инфицированных клеток. Значения параметров представлены в таблице 1.

Программная реализация

Когда операции перед собственным выполнением необходимы результаты выполнения другой операции, то между ними существует *зависимость* – основной лимитирующий фактор распараллеливания алгоритма. Наличие зависимостей может привести к невозможности распараллеливания программы. И наоборот, отсутствие зависимостей будет означать, что мы можем выполнять операции параллельно.

Класс проблем, для которых не требуется больших усилий, чтобы разделить их на ряд параллельных частей, называется «*embarrassingly parallel*» («потрясающе параллельный») [20]. Легкость распараллеливания обусловливается отсутствием зависимостей между этими параллельными частями. К такому классу проблем относятся: метод Монте-Карло [21], метод поиска полным перебором в криптографии или множество Мандельброта, где каждая точка может быть просчитана независимо. MPI (англ. *Message Passing Interface* – интерфейс передачи сообщений) отлично подходит для подобных задач, так как процессы не делят адресное пространство и память.

Однако далеко не все задачи могут быть легко распараллелены, в том числе многие типы искусственных нейронных сетей. В нейронных сетях с прямым распространением каждый нейрон связан с каждым, сигнал идет только в направлении от входного слоя к выходному. Подобные сети можно распараллеливать только по слоям. У слоистых нейронных сетей, наоборот, нейроны независимы от других на одном уровне, что дает возможность для распараллеливания, но не отдельных слоев, так как уровень L_n зависит от уровня L_{n+1}, который зависит от L_{n+2} и т.д. [22–25].

Иммунная сеть, которую мы моделируем в численном эксперименте, имеет такие же ограничения на распараллеливание, как и однослойная нейронная сеть с прямым распространением. Это динамическая сеть со сложными взаимодействиями на случайном графе, и в ней задача кластеризации невыполнима, так же как невозможна функциональная декомпозиция, которая осуществляется при моделировании климата [26].

В таких задачах возможность использования общей памяти (она имеет меньшую латентность по сравнению с динамической) является существенной для обеспечения оперативного доступа к требуемой информации, в данном случае к информации о численности других клонотипов при вычислении свободного места на антиген-

Таблица 2

	Однопоточная реализация		CUDA		OpenMP	
400 клонотипов	16 мс	6.7 час	1 мс	25 мин	3 мс	75 мин
900 клонотипов	47 мс	19.6 час	2 мс	50 мин	18 мс	7.5 час
2500 клонотипов	422 мс	7.3 дня	4 мс	1.6 час	156 мс	2.7 дня
5000 клонотипов	1797 мс	31.2 дня	7 мс	2.9 час	640 мс	11.1 дня
8100 клонотипов	5200 мс	90.2 дня	490 мс	8.5 дня	1875 мс	32.6 дня

Таблица 3

	Однопоточная реализация		CUDA		OpenMP	
400 клонотипов	19 мс	7.9 час	177 мс	3.1 дня	5 мс	2 час
900 клонотипов	67 мс	1.1 дня	185 мс	3.2 дня	32 мс	13.3 час
2500 клонотипов	562 мс	9.8 дня	199 мс	3.5 дня	170 мс	3 дня
5000 клонотипов	3162 мс	54.9 дня	398 мс	6.9 дня	1600 мс	27.8 дня
8100 клонотипов	15848 мс	275.1 дня	562 мс	9.7 дня	7700 мс	133.7 дня

презентующей клетке. Исходя из этого были выбраны технологии CUDA и OpenMP.

Известно, что для графических процессоров Nvidia (только такой тип находился в распоряжении авторов) технология OpenCL уступает CUDA по производительности [27].

Алгоритм метода Рунге–Кутты 4-го порядка для модели многоклональной сети Т-лимфоцитов с конкуренцией был написан на языке программирования C++ с использованием технологий CUDA и OpenMP. В качестве среды программирования использовалась среда Visual Studio 2010 Ultimate Edition, поддержка технологии CUDA осуществлялась с использованием CUDA Toolkit 4.2. Для веб-интерфейса программы использовался Apache-сервер, данные передавались через CGI в формате JSON с использованием концепции AJAX на основе JQuery.

Начальные значения численностей клонотипов эффекторных и наивных клеток, а также начальные пептидные концентрации во всех реализациях модели задавались одинаковым набором данных, кроме матриц пептидных связей K. Появление инфекции моделировалось только при достижении системой гомеостаза (на достижение такого состояния уходит половина модельного времени).

В таблице 2 представлено время, затраченное на одну итерацию, и общее время метода Рунге–Кутты с использованием кластера ННГУ им. Н.И. Лобачевского на CUDA (GPU – Nvidia Tesla M2070 [28], 448 ядер, 575 МГц), OpenMP (CPU – Intel Xeon L5630 [29] x4, 2.13 ГГц, многопоточная реализация) и для сравнения однопоточная реализация на этом же CPU.

В таблице 3 представлено время, затраченное на одну итерацию, и общее время метода Рунге–Кутты с использованием персонального компьютера на CUDA (GPU – Nvidia GeForce 9300M GS, 16 ядер, 450 МГц), OpenMP (CPU – Intel Pentium DualCore T4400 x2, 2.2 ГГц, многопоточная реализация) и для сравнения однопоточная реализация на этом же CPU.

Как видно из таблиц, использование GPU для численного интегрирования системы методом Рунге–Кутты (для 8100 клонотипов) позволило более чем в 3 раза уменьшить время расчета задачи на кластере и в 10 на персональном компьютере по сравнению с многопоточной реализацией на CPU. Однако стоит заметить, что для кластера технология CUDA всегда выигрывает по быстродействию, а для персонального компьютера – лишь при больших размерах системы (рис. 1, 2), из-за затрат по времени на выделение памяти и на передачу данных с ядра (*kernel*) на устройство (*device*). CUDA лучше там, где высока плотность арифметики, т.е. где число операций с памятью мало по сравнению с числом арифметических операций.

Для наглядности сравним время, затраченное на вычисления при однопоточной и многопоточной реализации для 8100 клонотипов с шагом интегрирования h=0.001 и временем интегрирования t=1500 (1.5×10^6 шагов интегрирования). На персональном компьютере для однопоточной реализации результат был бы получен примерно через 275 дней, для CUDA – через 10 дней, а на OpenMP – через 109 дней.

Заключение

Разработана программа для исследования математической модели популяционной динамики больших ансамблей клонотипов Т-клеток. Использование графического процессора для параллельных вычислений позволяет сильно сократить время счета по сравнению с многопоточной реализацией на центральном процессоре, что, в свою очередь, делает реальным анализ модели для длительных времен интегрирования с большим количеством клонотипов и пептидов. Большое количество клонотипов позволяет каче-

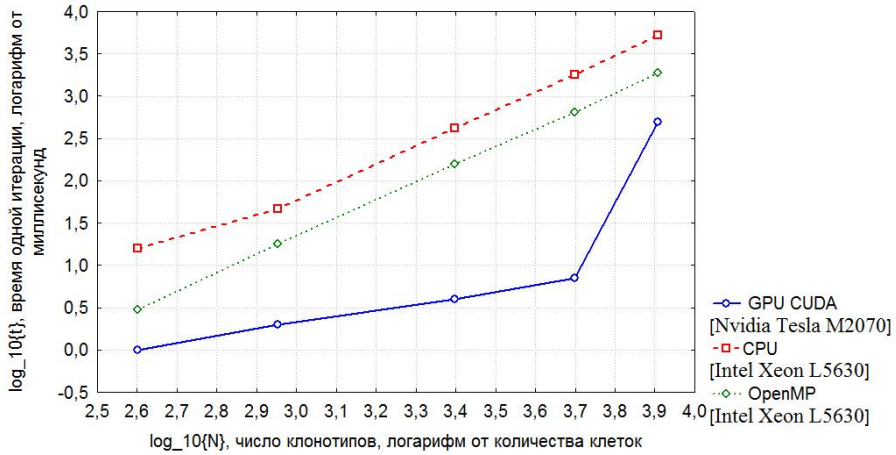

Рис. 1. Сравнение технологий CUDA, OpenMP и однопоточной реализации для численного эксперимента (кластер ННГУ им. Н.И. Лобачевского)

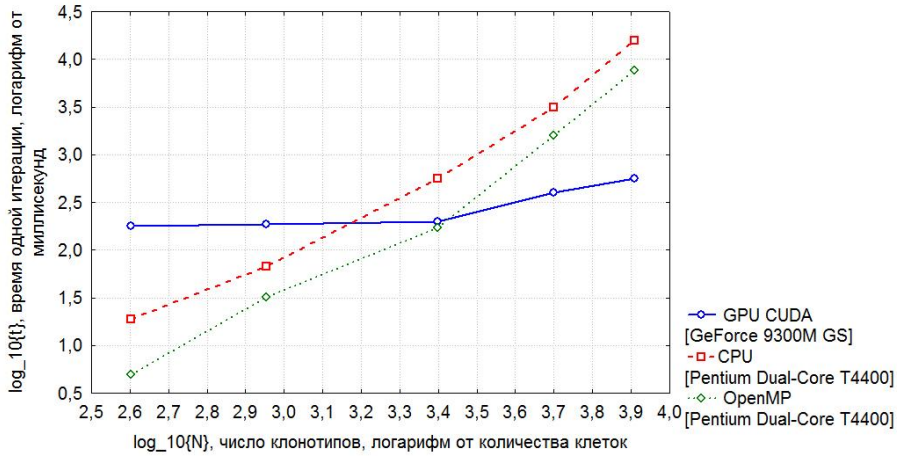

Рис. 2. Сравнение технологий CUDA, OpenMP и однопоточной реализации для численного эксперимента (персональный компьютер)

ственно обозначить задаваемое распределение для математической модели и получить статистически достоверное описание системы. Возможность численных расчетов с использованием таких более реалистичных моделей, в свою очередь, открывает дальнейшее поле для исследований, в том числе экспериментальных, предлагая верифицируемые предсказания о характере статистики пептидных концентраций и их изменениях при активации иммунопротеасомы.

Работа выполнена при поддержке ФЦП «Кадры», соглашение № 14.337.21.1234.

Список литературы

1. Hochberg R. Matrix Multiplication with CUDA. A basic introduction to the CUDA programming model. 2012. URL: http://www.shodor.org/media/content//petascale/materials/UPModules/matrixMultiplication/moduleDocument.pdf

2. Honghoon Jang. Neural Network Implementation Using CUDA and OpenMP. Digital Image Computing: Techniques and Applications (DICTA), 2008. P.155–161.

3. Kumar Aatish, Boyan Zhang. Three Dimensional Fast Fourier Transform CUDA Implementation. Department of Computer Science and Engineering, University of California, San Diego University of California, CSE260, 2012. URL: http://cseweb.ucsd.edu/~baden/classes/Exemplars/cse260_fa12/3DFFT.pdf

4. Sanders J., Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming. 2010. P. 312.

5. GPGPU.org. URL: http://www.gpgpu.org.

6. NVidia CUDA Programming Guide. URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

7. OpenCL. URL: www.khronos.org/opencl.

8. Спецификация OpenMP. URL: http://www.openmp.org/mp-documents/spec30.pdf.

9. Антонов А.С. Параллельное программирование с использованием технологии OpenMP: Учеб. пособие. М.: Издательство МГУ, 2009. 77 с.

10. Chapman B. et al. Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation). 2007. P. 384.

11. Perelson A., Weisbuch G. Immunology for physicists // Reviews of Modern Physics. 1997. V. 69. P. 1219.

12. Goldrath A.W., Bevan M.J. Selecting and maintaining a diverse T cell repertoire // Nature. 1999. V. 402. P. 255.

13. Jameson S.C. Maintaining the norm: T-cell homeostasis // Nature Reviews Immunology. 2002. V. 2. P. 547.

14. Peters J.M. et al. Distinct 19S and 20S subcomplexes of the 26S proteasome and their distribution in the nucleus and the cytoplasm // The Journal of Biological Chemistry. 2011. V. 269. P. 7709–7718.

15. Lalit Kumar Sharma et al. Activity-Based Near-Infrared Fluorescent Probe for LMP7: A Chemical Proteomics Tool for the Immunoproteasome in Living Cells // ChemBioChem. 2012. V. 13. P. 1899.

16. Sijts A.J. et al. MHC class I antigen processing of an adenovirus CTL epitope is linked to the levels of immunoproteasomes in infected cells // The Journal of Immunology. 2000. V. 164. P. 4500–4506.

17. De Boer R., Perelson A. Competetive control of self-renewing T cell repertoire // International Immunology. 1997. V. 9. P. 779.

18. Зорин Д.Ю., Иванченко М.В. Динамика иммунного ответа в многоклональных популяциях Т-клеток: роль иммуннопротеасомы // Вестник Нижегородского государственного университета. 2012. № 5(2). С. 92–99.

19. Or-Guil M. et al. Clonal Expansion of Cytotoxic T Cell Clones: the Role of the Immunoproteasome // Proceedings of the International Symposium on Mathematical and Computational Biology. 2005. P. 199.

20. Embarrassingly parallel. URL: http://en.wikipedia.org/wiki/Embarrassingly_parallel

21. Alexandrov V. et al. Parallel Monte Carlo Algorithms for Sparse. SLAE Using MPI // PVM/MPI'99, LNCS 1697. 1999. P. 283–290.

22. Fabrice Bernhard and Renaud Keriven. Spiking neurons on GPUs // International Conference on Computational Science (4). 2006. P. 236–243.

23. Jayram Moorkanikara Nageswaran et al. Efficient simulation of large-scale spiking neural networks using CUDA graphics processors // IJCNN '09. University of California, Irvine, 2009.

24. Udo Seiffert. Artificial neural networks on massively parallel computer hardware // Neurocomputing, 2004. 57:135–150. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.6.8003.

25. Uetz R., Behnke S. Large-scale Object Recognition with CUDA-accelerated Hierarchical Neural Networks // Proc. of ICIS. 2009.

26. Wehner M.F. Results From the parallel UCLA/LLNL atmospheric general circulation model // Proceedings of the 1st International AMIP Conference, Monterey, CA. 1995.

27. Kamran Karimi, Neil G. Dickson, Firas Hamze. A Perfomance Comparison of CUDA and OpenCL. arXiv:1005.2581v3. Retrieved 12 January 2012.

28. Nvidia Tesla M2070. URL: http://www.nvidia.com/docs/IO/43395/BD-05238-001_ v03.pdf.

29.Intel Xeon L5630. URL: http://ark.intel.com/products/47927/Intel-Xeon-Processor-L5630-12M-Cache-2_ 13-GHz-5_86-GTs-Intel-QPI.

MODELLING IMMUNE RESPONSE DYNAMICS IN T-CELL ENSEMBLES ON GRAPHIC PROCESSORS

D.Yu. Zorin, M.V. Ivanchenko

With the use of parallel computing technologies, a program has been developed for simulating the dynamics of T-lymphocyte multiclonal network model and immunoproteasome function. An analysis of some parallel computing methods is carried out. GPU technology has significantly reduced the duration of computations as compared with that on the central processor.

Keywords: competition, cellular network dynamics, mathematical immunology, parallel computing, CUDA, OpenMP.