Разработка высокопроизводительных методов моделирования эволюции бактериальных сообществ в программе «Гаплоидный эволюционный конструктор»

З.С.Мустафин*, Ю.Г.Матушкин, С.А.Лашин

Институт Цитологии и Генетики СО РАН

Новосибирский Государственный Университет

* e-mail Mustafinzs**@bionet.nsc.ru**

Введение

Моделирование эволюции бактериальных сообществ является актуальной задачей современной биоинформатики. Множество природных процессов происходит благодаря бактериям. Многие виды бактерий являются полезными для человека, а также играют важную роль в протекании процессов в окружающей среде. Научившись моделировать поведение и эволюцию бактериальных сообществ в тех или иных условиях, человечество сможет использовать это для развития современной медицины и науки.

Ранее в ИЦиГ СО РАН были разработаны методика моделирования и программный комплекс «Гаплоидный эволюционный конструктор» [1] (далее в тексте – ГЭК), осуществляющие моделирование эволюции прокариотических сообществ. ГЭК позволяет моделировать взаимодействия популяций между собой и с окружающей средой, включая различные эволюционно-популяционные процессы, такие как горизонтальный перенос генетического материала и мутации. ГЭК предназначен для моделирования сосуществования популяций трофически связанных одноклеточных гаплоидных организмов. Моделируются сообщества организмов, которые объединены в популяции по принципу генетической близости.

В работе приведены результаты разработки и программной реализации высокопроизводительного алгоритма моделирования популяционных процессов в рамках ГЭК: описан эффективный алгоритм, расчёта изменения численности популяции и параллельная реализация алгоритма с использованием технологии MPI [2]; проведены тестирование и оценка производительности алгоритма на высокопроизводительном кластере.

Разработка высокопроизводительного алгоритма расчета изменения численности популяции

В ходе работы был проведен анализ существующей версии ГЭК при помощи профилировщика Intel Parallel Amplifier (Intel Parallel Studio) [http://software.intel.com/ru-ru/articles/intel-parallel-studio-home] и было показано, что при моделировании сообществ с высоким генетическим разнообразием (10^6-10^8 уникальных аллельных комбинаций, что соответствует генетическому разнообразию крупных природных микробиологических сообществ) практически все время выполнения уходит на вычисление единственной функции – функции изменения численности популяции, независимо от типа трофической стратегии – функции расчета изменения численности популяции (рис. 1).

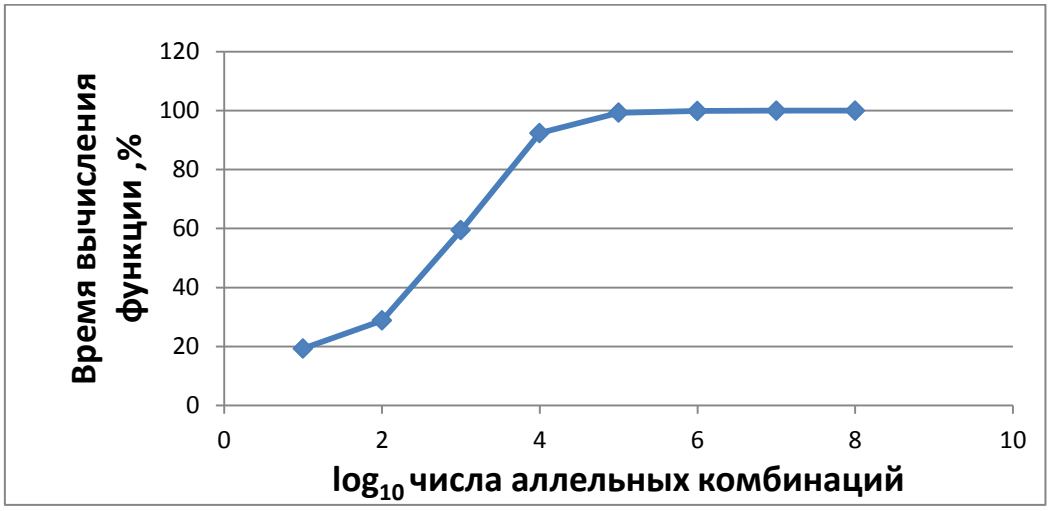

Рис. 1 График времени выполнения (в процентах от времени выполнения программы) функции изменения численности популяции при разном количестве аллельных комбинаций в моделируемом сообществе.

На рис.1 показано, что с ростом количества аллельных комбинаций время выполнения программы практически полностью концентрируется на функции изменения численности популяции. Опишем эту функцию более формально:

Признак – наличие процесса синтеза или утилизации какого-либо определённого субстрата. В рамках ГЭК каждый признак однозначно определяется одним геном. Ген рассматривается как единица наследования.

Аллель – как вариант гена, константа скорости синтеза или утилизации субстрата.

Обобщённый геном популяции в ГЭК – многомерное распределение частот аллелей для всех генов, присутствующих у особей популяции.

Например:

0(0.5) 1(0.2) 2(0.3) - распределение алеллей для гена 1

3(1) - распределение алеллей для гена 2

4(0.1) 5(0.1) 6(0.4) 7(0.4) - распределение алеллей для гена 3

8(0.9) 9(0.1) - распределение алеллей для гена 4

Рис. 2 Пример представления обобщённого генома популяции в ГЭК.

На рисунке 2 показано 4 гена с 3,1,4 и 2 возможными вариантами аллелей и с соответствующими каждому варианту частотами (концентрация аллельного варианта). Для расчёта прироста численности популяции с учётом различной приспособленности особей, несущих разные аллельные комбинации, необходимо рассмотреть все возможные такие комбинации и учесть в каждой из них изменение размера популяции с помощью заданной пользователем трофической стратегии, а затем посчитать итоговый размер популяции и концентрацию каждого аллельного варианта.

В работе был реализован следующий подход: набор генов представляется в виде контейнера, в который последовательно записываются все варианты аллелей каждого гена, а так же запоминается индекс первого аллельного варианта каждого гена. Если рассмотреть пример на рис. 2, то полученные объекты будут иметь вид:

(**0**, 1, 2, **3**, **4**, 5, 6, 7, **8**, 9) – значения аллельных вариантов

(**0.5**, 0.2, 0.3, **1**, **0.1**, 0.1, 0.4, 0.4, **0.9**, 0.1) – концентрации алелелей

(0,3,4,8) – массив индексов (позиций в текущей комбинации) генов.

Полужирным шрифтом выделены индексы стартовых (текущих) аллельных вариантов каждого гена.

Выписывается первая комбинация (0,3,4,8), затем проверяется, является ли текущий вариант аллеля последнего гена последним из возможных вариантов аллеля. Если не является, идет увеличение индекса текущего аллельного варианта последнего гена на единицу (в позицию (0,3,4,9)). Как только будет получен последний аллельный вариант последнего гена, то индекс текущего варианта предыдущего гена увеличивается на 1, а индекс текущего варианта последнего гена возвращается в начало (в итоге получена позиция (0,3,5,8)). Если текущие варианты двух последних генов являются последними, то у обоих генов текущими становятся первые аллельные варианты, а у идущего перед ними гена выбирается следующий вариант.

Процесс повторяется, пока не будут пройдены все возможные комбинации аллелей.

Например, первые пять итераций дадут следующие комбинации:

(**0**, 1, 2, **3**, **4**, 5, 6, 7, **8**, 9)		(0, 3, 4, 8)
(**0**, 1, 2, **3**, **4**, 5, 6, 7, 8, **9**)		(0, 3, 4, 9)
(**0**, 1, 2, **3**, 4, **5**, 6, 7, **8**, 9)	=>	(0, 3, 5, 8)
(**0**, 1, 2, **3**, 4, **5**, 6, 7, 8, **9**)		(0, 3, 5, 9)
(**0**, 1, 2, **3**, 4, 5, **6**, 7, **8**, 9)		(0, 3, 6, 8)
Развертка значений аллельных вариантов		Комбинация

Достоинство данного алгоритма состоит в том, что его можно эффективно распараллелить на любое количество процессов (испытания проводились до 900 процессов, но, исходя из общих соображений, мы предполагаем, что алгоритм будет работать и для значительно большего числа процессов). Чтобы алгоритм работал корректно, всем процессам необходимо передать развертки значений и концентраций. Передача разверток осуществляется с помощью функции массовой рассылки сообщений (в MPI это MPI_BCast). Аналогично осуществляется передача всех параметров, необходимых для вычисления прироста численности популяций (параметры трофических стратегий). Далее каждый процесс вычисляет свою стартовую и конечную позиции (на рис. 2 нулевой процесс начинает с комбинации (0,3,4,8), остальные сами вычисляют индексы своих стартовых и конечных позиций).

Реализованная схема показана на рис. 3. Суммирование реализовано с помощью функции MPI_Reduce. После завершения работы все вспомогательные процессы удаляют динамические объекты и сразу завершают работу. Корневой процесс присваивает концентрациям аллелей и размеру популяций значения, полученные в процессе выполнения, удаляет динамически выделенные объекты и так же завершает работу.

Результаты тестовых расчетов

В рамках работы было проведено тестирование последовательных версий и параллельной версии на 4-х ядерных процессорах E5540 2.53 GHz (Nehalem) и 6-ти ядерных процессорах X5670 2.93 GHz (Westmere) кластера НКС 30-Т [http://bioinformatics.bionet.nsc.ru]. Алгоритм был верифицирован на тестовом наборе сценариев ГЭК. Затем были составлены специальные нагрузочные тесты (10^6-10^8 аллельных комбинаций). Расчёт нагрузочных

тестов с помощью последовательных версий на процессорах E5540 (2.53 GHz) показал выигрыш в скорости выполнения для нового алгоритма чуть более 3 часов, а на процессорах X5670 (2.93 GHz) 2 часа 20 минут (для случая 10^8 аллельных комбинаций). Полные результаты показаны на графиках (рис.4-5).

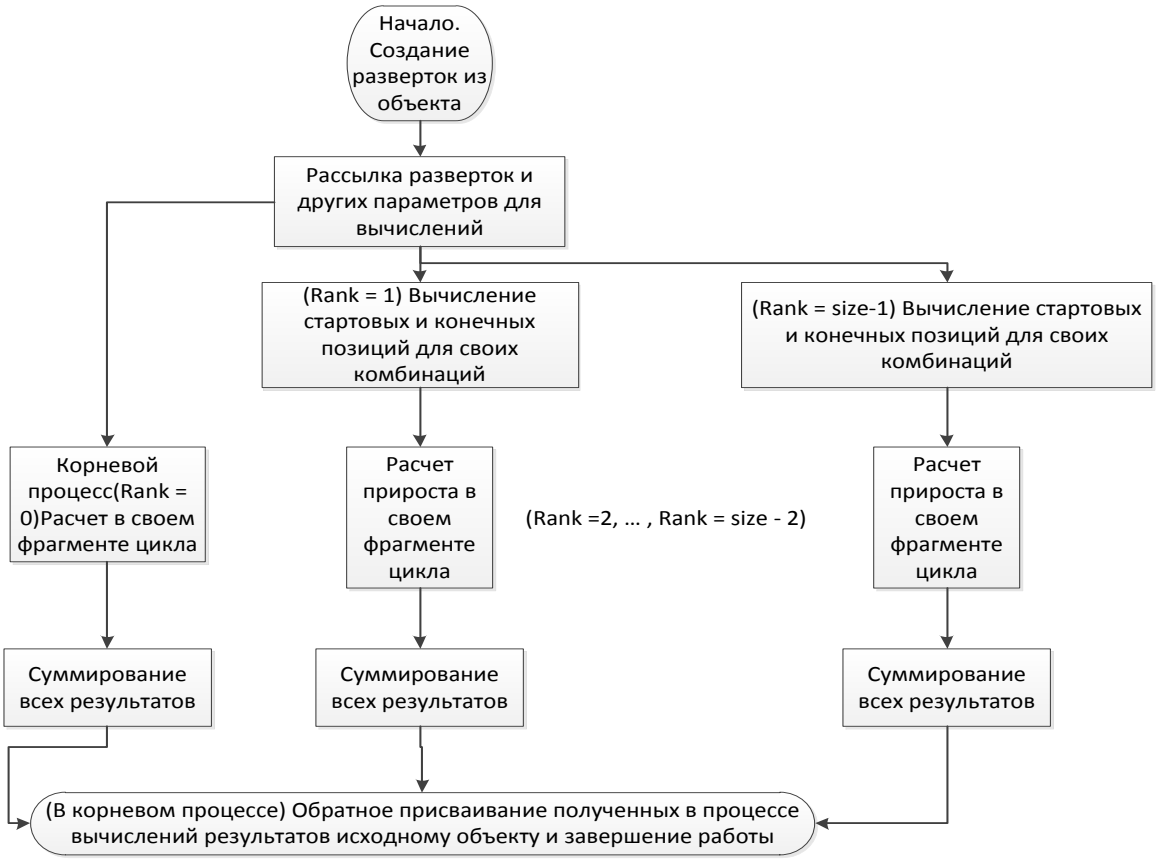

Рис. 3 Схема распараллеливания алгоритма.

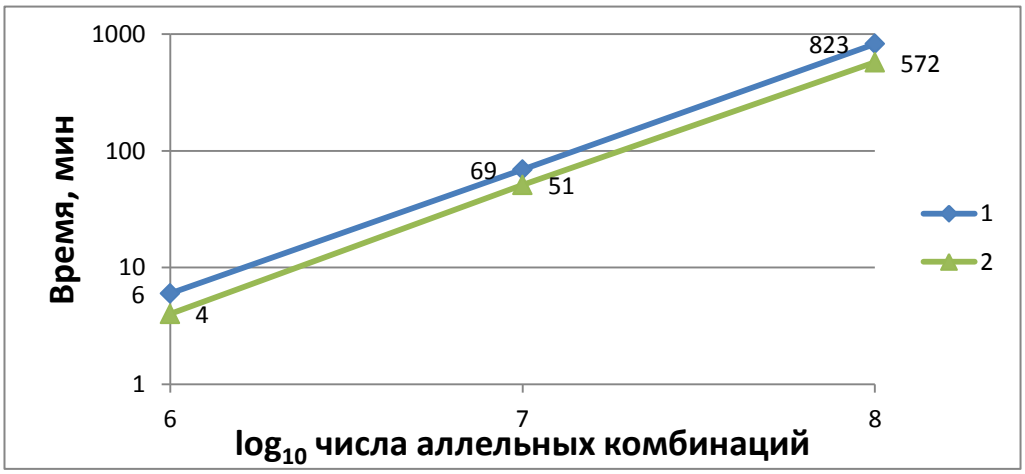

Рис. 4 Время выполнения первоначальной (1) и предложенной в работе (2) последовательных версий на процессорах E5540.

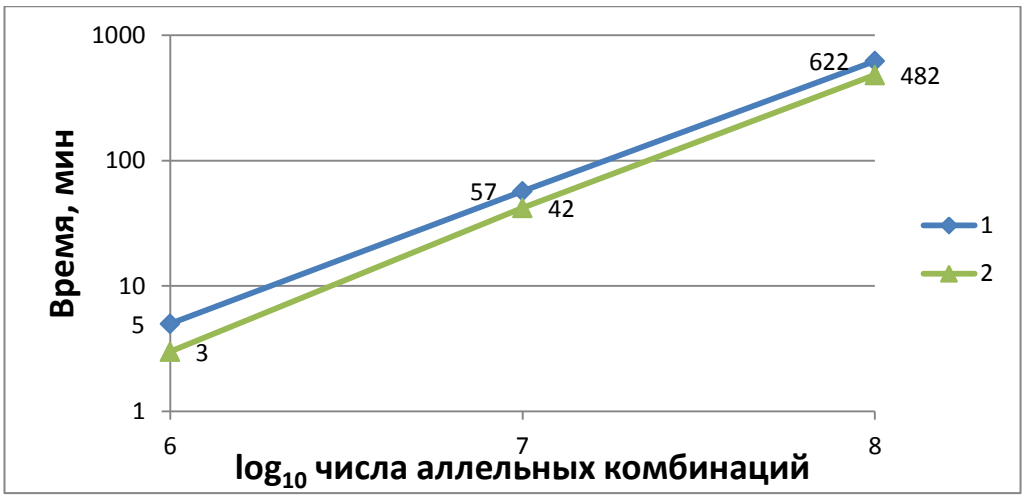

Рис. 5 Время выполнения первоначальной (1) и предложенной в работе (2) последовательных версий на процессорах X5670.

Результаты тестирования параллельной версии алгоритма приведены в таблицах 1 - 2.

Таблица 1 Результаты тестирования на E5540.

Кол-во процессов	Время выполнения	Эффективность распараллеливания	Ускорение	Дисперсия, сек	Время ожидания
1	9:32:25	1	1	533	0:00:00
2	4:43:39	1.0090	2,018	508	0:00:00
4	2:21:51	1.0088	4,0352	105	0:00:00
8	1:10:57	1.0084	8,0672	1	2:56:28
16	0:35:25	1.0101	16,1616	1	7:39:44
24	0:24:28	0.9748	23,3952	163	5:28:19
32	0:17:55	0.9984	31,9488	1	8:03:16
64	0:09:02	0.9901	63,3664	2	4:32:56

В таблице 1 можно увидеть, что эффективность распараллеливания может быть немного больше 1. Это связано с тем, что на разных узлах кластера могут быть получены разные значения времени выполнения программы. Среднеквадратичное отклонение, с округлением до секунд, показано в столбце «Дисперсия».

Также было посчитано время ожидания очереди. Столбец «Время ожидания» показывает, что иногда лучше запросить меньше процессов и подождать несколько минут, чем запросить много процессов и часами ждать очереди.

Таблица 2 Результаты тестирования на X5670.

Кол-во процессов	Время выполнения	Эффективность распараллеливания	Ускорение	Дисперсия, сек	Время ожидания
1	8:02:26	1	1	249	0:00:00
2	4:10:24	0.9633	1,9266	65	0:00:00
4	2:05:20	0.9623	3,8492	188	0:00:00
8	1:02:03	0.9718	7,7744	4	0:00:00
16	0:31:06	0.9695	15,512	3	0:00:00
24	0:20:48	0.9664	23,1936	3	0:00:00
36	0:13:46	0.9734	35,0424	1	0:00:11
64	0:07:52	0.9582	61,3248	2	7:21:43
96	0:05:13	0.9633	92,4768	1	13:19:49
144	0:03:32	0.9481	136,5264	1	96:43:25
264	0:02:03	0.8914	235,3296	2	6:17:39

Таким образом, время выполнения программы на современных процессорах сократилось с 10.5 часов до 14 минут, с учетом времени ожидания, при этом требуя всего 3 узла вычислительного комплекса и на каждый параллельный процесс необходимо всего 2 Мб оперативной памяти.

Значительное ускорение выполнения программы позволяет пользователю увеличить сложность и разнообразие моделируемых биологических ситуаций, что способствует развитию эволюционной биологии.

Работа частично поддержана грантом РФФИ 12-07-00671-а, Инт. Проектом СО РАН №47.

Литература

[1] Lashin S.A., Suslov V.V., Kolchanov N.A., Matushkin Yu.G. Simulation of coevolution in community by using the "Evolutionary Constructor" program. In Silico Biology, 2007, V 7, N 3, 261-275.

[2] Корнеев В.Д., Параллельное программирование в MPI, 2-е изд., испр. – Новосибирск: ИВМиМГ СО РАН 2002. – 215 с.