© 2012 г. М.А. МАРЧЕНКО, канд. физ.-мат. Наук
(Федеральное государственное бюджетное учреждение науки
Институт вычислительной математики и математической геофизики
Сибирского отделения Российской академии наук, Новосибирск)

ТЕХНОЛОГИИ РАСПРЕДЕЛЕННОГО СТАТИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ НА СУПЕР ЭВМ[1]

В работе представлена технология распределенного статистического моделирования с использованием библиотеки *PARMONC* (сокращение от "PARallel MONte Carlo"), предназначенной для эффективного распараллеливания широкого круга приложений метода Монте-Карло, обладающих большой вычислительной трудоемкостью. При распараллеливании используется «естественная» крупноблочная фрагментированность алгоритмов метода Монте-Карло. "Ядром" библиотеки является тщательно протестированный, быстрый и надежный длиннопериодный параллельный генератор псевдослучайных чисел. Библиотека представляет собой простой в использовании программный инструмент для организации распределенных вычислений, не требующий от пользователя знания языка MPI; распараллеливание сложных последовательных программ статистического моделирования производится достаточно просто. Обсуждаются проблемы масштабирования распределенного статистического моделирования на большое число ядер.

TECHNOLOGIES OF THE DISTRIBUTED STOCHASTIC SIMULATION ON SUPERCOMPUTERS / M.A. Marchenko (Institute of Computational Mathematics and Mathematical Geophysics, Prospect Lavrentieva 6, Novosibirsk 630090, Russia, E-mail: marchenko@sscc.ru)
In this paper, the technology of the distributed stochastic simulation is presented. The technology is based on the use of the software library *PARMONC* (an acronym of the PARallel MONte Carlo) that was developed for the massively parallel simulation by the Monte Carlo method on supercomputers is presented. Native coarse-grain parallelism of the Monte Carlo method is used in the library. A "core" of the library is a well-tested, fast and reliable long-period parallel pseudorandom numbers generator. The *PARMONC* is an easy-to-use program instrument to perform distributed stochastic simulation. Routines from the *PARMONC* can be called in the user-supplied programs without explicit usage of MPI instructions. Parallelization of complicated sequential Monte Carlo programs can be made in an easy way. Problems of scaling the stochastic simulation to many computational cores are discussed.

[1] Работа проводилась при финансовой поддержке грантов РФФИ №№ 12-01-00727, 12-01-00034, 10-07-00454; МИП № 39, 47, 126, 130 СО РАН.

1. Введение

В настоящее время ведущими специалистами по вычислительной математике высказывается убеждение в том, что в ближайшем будущем в области компьютерного моделирования будут широко использоваться вероятностные имитационные модели и методы Монте-Карло (методы численного статистического моделирования). С одной стороны, это убеждение основано на том, что вероятностные имитационные модели дают адекватное описание физических, химических, биологических и др. явлений при их рассмотрении «из первых принципов». С другой стороны, алгоритмы метода Монте-Карло, реализующие вероятностные модели, допускают возможность эффективного распараллеливания. Использование методов Монте-Карло весьма перспективно в связи с вероятным появлением в ближайшем будущем супер-ЭВМ экзафлопсного уровня [1].

Упомянем некоторые из важных применений методов статистического моделирования. С течением химически реагирующих газов часто приходится сталкиваться при рассмотрении физических явлений, связанных с повышением температуры газовых смесей выше критической. К таким явлениям относятся процессы горения газовых топлив в камерах сгорания, соплах и горелках различных геометрических форм. Численное моделирование на компьютерах течений химически реагирующих газов по методу Монте-Карло является весьма трудоёмким, поскольку приходится учитывать многие физические и химические особенности протекающих при горении процессов. Как показывает анализ литературы, в последние десятилетия понимание химии горения (по крайней мере, с участием небольших молекул) и возможности моделирования процессов горения на больших компьютерах достигли такого уровня, который обеспечивает необходимую надежность результатов [2]. Численное моделирование процессов горения требует значительных вычислительных ресурсов и относятся к числу задач, для которых необходимо использование супер-ЭВМ. Такие вычислительные потребности обусловлены большим средним временем расчета каждой реализации ансамбля тестовых частиц, большим числом независимых реализаций ансамбля, которое определяется из необходимой точности расчетов, и большим объемом используемой машинной памяти.

Еще одним примером «большой» вычислительной задачи является моделирование эмиссии, размножения и транспорта электронов в электрическом поле в газовой среде по методу индивидуальных соударений [3]. Задача такого рода в настоящее время рассматривается в рамках выполнения междисциплинарного интеграционного проекта СО РАН №126, в число исполнителей которого входит автор настоящей работы. Как показывают расчеты, в каждой реализации число вторичных электронов (полученных от одного первичного) варьируется в пределах от нескольких сотен тысяч до нескольких миллионов. При этом время моделирования реализаций на кластерах ССКЦ КП СО РАН варьируется в пределах от нескольких часов до десятков часов.

Для такого рода задач применение технологии распределенного статистического моделирования, предлагаемой в настоящей работе, позволит существенно сократить трудоемкость расчетов.

2. Распределенное статистическое моделирование

Под численным статистическим моделированием обычно понимают реализацию с помощью компьютера вероятностной модели ζ некоторого объекта с целью оценивания изучаемых интегральных характеристик φ на основе закона больших чисел [4]:

$$\varphi \approx \mathrm{E}\zeta \approx \bar{\zeta} = \frac{1}{L}\sum_{i=1}^{L}\zeta^{(i)} .$$

При этом, чем больше объем выборки L, составленной из смоделированных независимых реализаций $\zeta^{(1)},\zeta^{(2)},...,\zeta^{(L)}$, тем выше точность оценивания, причем статистическая погрешность убывает обратно пропорционально квадратному корню от объема выборки L.

Трудоемкость статистического моделирования (т.е. затраты машинного времени, необходимые для достижения заданного уровня статистической погрешности ε) определяется величиной $\tau_\zeta L$, которая, в свою очередь, пропорциональна величине $\tau_\zeta D\zeta\varepsilon^{-2}$. Здесь τ_ζ - среднее время моделирования одной реализации случайной величины ζ, $D\zeta$ - дисперсия случайной величины ζ. «Большие» задачи статистического моделирования характеризуются либо большой величиной τ_ζ, либо малой величиной ε, либо большой величиной $D\zeta$, либо же сочетанием этих факторов. Кроме того, «большие» задачи статистического моделирования часто требуют больших объемов машинной памяти.

С целью понижения трудоемкости можно применять распараллеливание статистического моделирования [5-7]. В зависимости от условий задачи и параметров алгоритма статистического моделирования применяются разные методики параллельной реализации.

Следует упомянуть класс приложений, например, методы прямого статистического моделирования пространственно неоднородных задач газовой динамики и теории дисперсных систем, для которых ресурсов одного вычислительного ядра недостаточно для моделирования отдельных реализаций (реализацией здесь является ансамбль тестовых частиц) и требуется применять методы пространственной декомпозиции ансамбля. При этом можно применять способы динамической балансировки вычислительной нагрузки, основанные на специальных формулах прогноза времени вычислений для каждого вычислительного ядра [6].

Метод *распределенного статистического моделирования* состоит в распределении моделирования независимых реализаций по вычислительным ядрам с периодическим осреднением полученных выборочных значений по статистически эффективной формуле

$$\bar{\zeta} = \left[\sum_{m=1}^{M} l_m\right]^{-1} \sum_{m=1}^{M} l_m \bar{\zeta}_m , \qquad (1)$$

где M - общее число ядер, l_m - объем выборки, полученной на m-м ядре, $\bar{\zeta}_m$ - соответствующее m-му ядру выборочное среднее значение. Очевидно, что главным критерием осуществимости такой параллельной реализации является возможность «поместить» данные вычислительной программы для моделирования реализаций в оперативную память каждого ядра. Подчеркнем, что в целях распределенного статистического моделирования допустимо использовать вычислительные ядра с разной производительностью [5]. При этом обмен данными можно свести к минимуму, допуская только начальную «загрузку» вычислительных ядер и финальное получение выборочных средних. Действуя таким образом, можно добиться обратно пропорциональной зависимости величины трудоемкости «распределенной» случайной оценки (1) от числа ядер, при условии, что используемые ядра имеют одинаковую производительность. Отметим, что возможные отказы компонент вычислительной системы и связанные с этим потери данных компенсируется моделированием реализаций на других вычислительных узлах [5].

Описанную методику распределенного статистического моделирования можно легко модифицировать с целью эффективной оценки функционалов, зависящих от параметра $x \in X$:

$$\varphi \approx \mathrm{E}\zeta(x;\omega).$$

При этом моделируемое распределение случайной величины ω может зависеть, а может и не зависеть от параметра x. Аналогичную методику можно также использовать для оценки функционалов, возникающих в результате осреднения базовых функционалов по случайному параметру σ задачи, например по случайной плотности среды. Вероятностное представление при этом имеет вид двойного математического ожидания

$$\varphi \approx \mathrm{EE}\zeta(\omega,\sigma).$$

Примеры подобных задач приведены в [5].

Как правило, при параллельной реализации необходимый объем выборки базовых случайных чисел очень велик, поэтому целесообразно использовать длиннопериодные псевдослучайные последовательности. А именно, для решения «больших» задач по методу Монте-Карло предлагается использовать генератор следующего вида [5, 7]:

$$u_0 = 1, u_n \equiv u_{n-1} A (\bmod 2^{128}), \alpha_n = u_n 2^{-128}, \mathrm{n} = 1,2,\ldots, \quad (2)$$

где

$$A \equiv 5^{100109} (\bmod 2^{128}).$$

Последовательность псевдослучайных чисел $\{\alpha_n\}$ является периодической, длина ее периода равна $2^{126} \approx 10^{38}$.

Для распределения псевдослучайных чисел между разными вычислительными ядрами предлагается следующий порядок их использования. Последовательность $\{\alpha_n\}$ предварительно разбивается на подпоследовательности длины μ, начинающиеся с чисел $\{\alpha_{\mathrm{m}\mu}\}$, $\mathrm{m} = 0,1,2,\ldots$, после чего разные подпоследовательности используются на разных ядрах. Значение «прыжка» генератора μ должно выбираться из соображений, чтобы μ псевдослучайных чисел хватало для моделирования на каждом ядре. Легко показать, что для метода вычетов начальные значения $\{\alpha_{m\mu}\}$ указанных подпоследовательностей получаются по формуле

$$u_{(m+1)\mu} = u_{m\mu} A_\mu (\bmod 2^{128}), \alpha_{m\mu} = u_{m\mu} 2^{-128}, \mathrm{m} = 0,1,\ldots \quad (3)$$

$$A_\mu \equiv A^\mu (\bmod 2^{128}).$$

Отметим, что длина периода генератора позволяет независимым образом распределять псевдослучайные числа по реализациям на практически неограниченное число вычислительных ядер. Параллельный генератор (3) успешно используется в ряде институтов СО РАН на протяжении последних десяти лет.

3. Реализация распределенного статистического моделирования с помощью библиотеки PARMONC

С целью унификации применения распределенного статистического моделирования при решении широкого круга задач методом Монте-Карло разработана и внедрена в широкое использование программная библиотека *PARMONC* (сокращение от *PARallel MONte Carlo*) [10]. Область применения библиотеки: «большие» задачи статистического моделирования в естественных и гуманитарных науках (физика, химия, биология, меди-

цина, экономика и финансы, социология и др.) Библиотека *PARMONC* установлена на кластерах Сибирского суперкомпьютерного центра (ССКЦ КП СО РАН) и может использоваться на вычислительных системах с аналогичной архитектурой. При этом использование библиотеки не привязано к каким-то определенным компиляторам языков C и FORTRAN или MPI. Инструкции по использованию библиотеки с примерами можно найти по ссылке [9].

Возможность применения библиотеки *PARMONC* определяется «естественной» крупноблочной фрагментированностью программ статистического моделирования. Общая структура такого рода программ, в упрощенном виде, следующая (в нотации языка C++):

```
void main( void ) {
        long int i, L;
        TypeRL RL, SUBT;
        SUBT = 0.0;
        //      цикл по реализациям
        for( i = 0; i < L; i++ ) {
                // далее идут операторы, вычисляющие реализацию RL
                ….
                SUBT= SUBT + RL;
        }
        SUBT = SUBT / L;
}
```

Здесь *L* – общее число независимых реализаций случайного объекта, задаваемых композитным типом данных *TypeRL* (допускается поэлементное суммирование переменных такого типа); реализации *RL* моделируются внутри цикла по переменной *i*. Полученные таким образом реализации *RL* (статистически независимые в совокупности) добавляются к «счетчику» *SUBT* и по выходу из цикла осредняются, что дает статистическую оценку искомого математического ожидания случайного объекта.

При распараллеливании последовательных программ с помощью *PARMONC* определяется процедура *realization* (моделирующая подпрограмма), возвращающая одну реализацию *RL* (возвращение - через аргумент процедуры). При этом считается, что моделирующая подпрограмма использует потоки псевдослучайных чисел, генерируемых внешней по отношению к ней подпрограммой. Взаимосвязь программных объектов при статистическом моделировании поясняется на рис. 1.

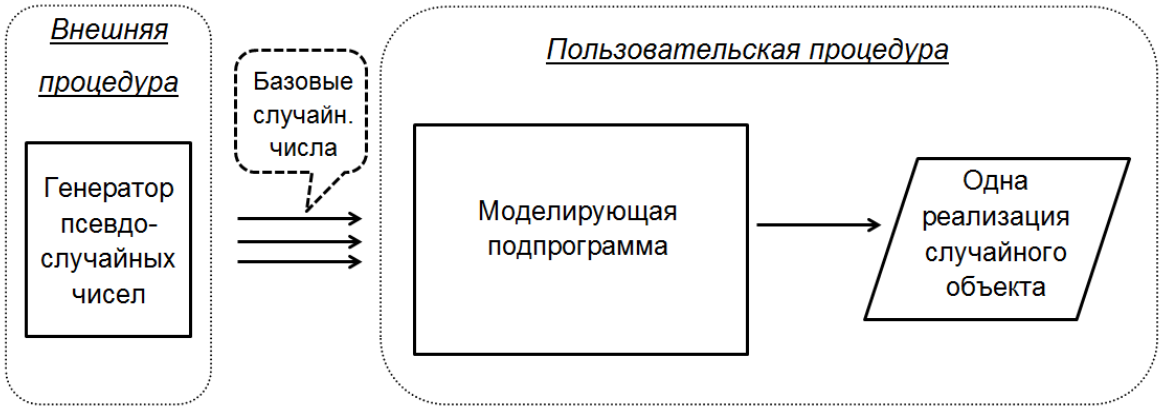

Рис. 1. Взаимосвязь программных объектов при статистическом моделировании.

С учетом такой взаимосвязи цикл по независимым реализациям и финальное осреднение заменяются вызовом библиотечной процедуры следующего вида:

parmoncc(realization, L, SUBT , …);

Здесь имя моделирующей подпрограммы и общее число независимых реализаций передаются в библиотечную процедуру *parmoncc* в качестве входных аргументов; выборочное среднее будут возвращаться в переменную *SUBT*; для простоты остальные аргументы процедуры *parmoncc* пока опущены и заменены многоточием. Процедура *parmoncc* автоматически распределяет моделирование независимых реализаций по вычислительным ядрам. Все остальные операторы пользовательской программы остаются без изменений. Таким образом, в итоге имеем следующий код, пригодный для компиляции и сборки с помощью библиотеки *PARMONC*:

```
void main( void ) {
        TypeRL SUBT;
        parmoncc (realization, L, SUBT , … );
}
void realization( TypeRL  RL ){
        //      далее идут операторы моделирующей подпрограммы
        …
        //      вычисленная реализация возвращается в переменную RL
}
```

В процессе распределенных вычислений на каждом ядре используются потоки независимых псевдослучайных чисел, получаемые в результате работы подпрограммы, реализующей параллельный генератор (3). В процедуре *realization* библиотечный параллельный генератор вызывается следующим образом (см. следующий раздел):

a = rnd128();

Здесь *a* – очередное псевдослучайное число, равномерно распределенное в интервале от нуля до единицы. Инициализация параллельного генератора производится автоматически

при запуске программы, скомпилированной и собранной с помощью библиотеки *PARMONC*.

При использовании параллельного генератора в *PARMONC* используется иерархия вложенных подпоследовательностей базовой последовательности:

- внутри базовой последовательности $\{\alpha_n\}$ делаются "прыжки" длиной n_e для определения начальных элементов подпоследовательностей, назначаемых разным случайным экспериментам,
- внутри каждой подпоследовательности для экспериментов делаются "прыжки" длиной $n_p < n_e$ для определения начальных элементов подпоследовательностей, назначаемых разным процессорам,
- внутри каждой подпоследовательности для процессоров делаются “прыжки” длиной $n_r < n_p$ для определения начальных элементов подпоследовательностей, назначаемых разным реализациям.

Номер подпоследовательности для экспериментов определяется пользователем; номера подпоследовательностей для процессоров и реализаций определяются в *PARMONC* автоматически. Используемая иерархия поясняется на рис. 2.

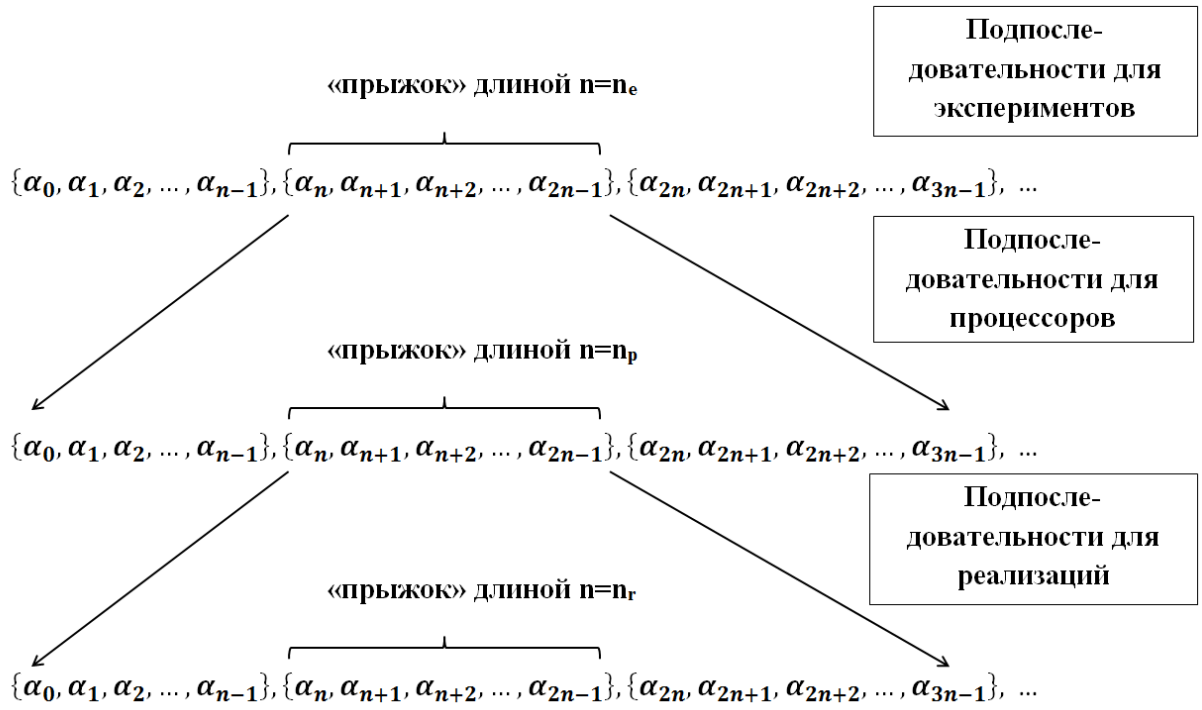

Рис. 2. Иерархия вложенных подпоследовательностей в базовой последовательности генератора псевдослучайных чисел (3).

По умолчанию, длины «прыжков» следующие:

- $n_e = 2^{115} \approx 10^{34}$ - в подпоследовательности для экспериментов,
- $n_p = 2^{98} \approx 10^{29}$ -в подпоследовательности для процессоров,
- $n_r = 2^{43} \approx 10^{13}$ - в подпоследовательности для реализации.

Таким образом, по умолчанию, с помощью *PARMONC* можно сделать следующее*:*

- провести $2^{125} \cdot 2^{-115} \approx 10^3$ экспериментов,
- в рамках одного эксперимента "загрузить" $2^{115} \cdot 2^{-98} \approx 10^5$ процессоров,
- на одном процессоре смоделировать $2^{98} \cdot 2^{-43} \approx 10^{16}$ реализаций.

Соответствующие множители для параллельного генератора (3) для выполнения "прыжков" можно переопределить с помощью библиотечной команды *genparam*.

Таким образом, библиотечные подпрограммы применяются без явного использования команд MPI. Поскольку период параллельного генератора (3) достаточно большой, то число используемых в целях моделирования в *PARMONC* вычислительных ядер практически не ограничено и зависит только от используемой вычислительной системы.

Для достижения равномерной загрузки ядер распределенное статистическое моделирование организовано следующим образом. Каждое ядро (например, с номером m, где $m = 0,1,2,...,M-1$) независимо моделирует реализации $\zeta^{(1)},\zeta^{(2)},...$ и периодически, скажем, через каждые *LS* реализаций, дает команду на отправку вычисленных выборочных средних $\bar{\zeta}_m$ на выделенное ядро (с номером 0, для определенности).

В свою очередь, 0-е ядро периодически, скажем, через каждые *LR* реализаций, дает команду на получение отправленных с других ядер значений, осредняет их по формуле (1) и сохраняет на диск. Отметим, что отправка данных со всех ядер на 0-е ядро и получение 0-м ядром отправленных ему данных происходит в асинхронном режиме. Значения периодов отправки *LS* и приемки *LR* задаются в числе параметров процедуры *parmoncc*:

parmoncc(realization, L, SUBT , LR, LS).

При такой организации вычислений параметр *L* целесообразно задавать достаточно большим, а параметры отправки *LS* и приемки *LR* определять так, чтобы результаты осреднения на 0-м процессоре обновлялись на диске через удобные для пользователя промежутки времени. Такая организация вычислений позволяет пользователю контролировать погрешность моделирования. Организация вычислений схематически представлена на рис. 3 для числа ядер $M = 4$.

Как показали численные эксперименты с использованием *PARMONC*, при использовании большого числа ядер, при больших объемах пересылаемых данных и при большой частоте отправки/приемки, асинхронная процедура передачи данных в процессе счета практически не влияет на эффективность распараллеливания, величина которой составляет почти 100% [10].

В процессе счета происходит автоматическое вычисление выборочных средних и границ погрешностей для статистических оценок, алгоритм моделирования которых задается в моделирующей подпрограмме; результаты вычислений периодически сохраняются на диске в удобном для дальнейшей обработки виде. С помощью библиотеки *PARMONC* можно легко организовать продолжение ранее проведенных расчетов с автоматическим учетом их результатов. Также с помощью библиотеки можно получать коррелированные статистические оценки различных функционалов [9, 10].

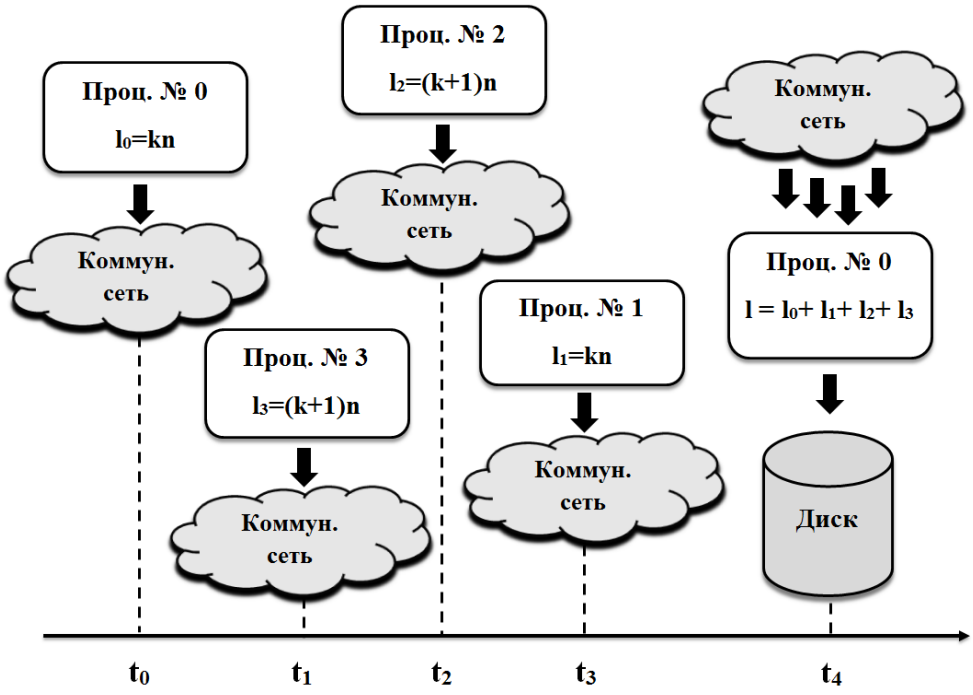

Рис. 3. Схема организации вычислений в PARMONC (для числа ядер $M = 4$).

4. Особенности реализации библиотеки PARMONC на кластере ССКЦ КП СО РАН.

Оборудование кластера НКС-30Т ССКЦ КП СО РАН позволяет производить длительные по времени расчеты [11]. Для моделирования реализаций в соответствии с методологией распределенного статистического моделирования на каждом из вычислительных ядер с MPP архитектурой доступно 2 Гб или 8 Гб оперативной памяти в зависимости от используемых серверов кластера. Коммуникационная сеть кластера позволяет пересылать между вычислительными ядрами данные большого объема (данные соответствуют рассчитанным на ядрах выборочным значениям). С целью апробации библиотеки моделирование реализаций распределялось на все ядра кластера, общее число которых составляет почти 3000. Таким образом, на кластере НКС-30Т можно эффективно производить «большие» расчеты по методу Монте-Карло.

На кластере НКС-30Т библиотека *PARMONC* установлена в директории */ifs/apps/parmonc/*. Библиотека состоит из следующих подпрограмм и исполняемых файлов [9, 10]:

- *rnd128* - функция для получения одного случайного числа, равномерно распределенного в интервале от 0 до 1, с помощью параллельного генератора случайных чисел (3);
- *parmoncf* - процедура, осуществляющая распределенное статистическое моделирование (для программ на Fortran-е);
- *parmoncc* - процедура, осуществляющая распределенное статистическое моделирование (для программ на C);

- *manaver* - программа для осреднения выборочных средних, независимо рассчитанных на разных ядрах;
- *genparam* - программа для расчета параметров параллельного генератора случайных чисел.

Здесь *rnd128, parmoncf* и *parmoncc* являются библиотечными подпрограммами для использования в пользовательских программах на Fortran-е, C или C++, *genparam* и *manaver* являются исполняемыми файлами для запуска из командной строки. Объектные файлы *PARMONC* упакованы в статическую библиотеку *libparmonc.a*.

Пользователь вставляет вызовы процедур *rnd128* и *parmoncf/parmoncc* в свои программы. Главная пользовательская программа, в которой находится вызов *parmoncf/parmoncc,* рассматривается компилятором как MPI-программа, несмотря на то, что в самой пользовательской программе нет явных вызовов директив MPI. Это означает, что такая программа должна компилироваться и собираться с использованием команд *mpiicc* или *mpiifort*. Результаты расчетов, выполненных с использованием процедур *parmoncf/parmoncc*, сохраняются в файлы, которые находятся в специальной поддиректории в рабочей директории пользователя. Все эти файлы обновляются всякий раз, когда выделенное ядро (например, 0-е, для определенности) получает данные с других ядер, осредняет их и сохраняет на диск.

В файл *.bashrc*, который находится в домашней директории пользователя, необходимо добавить следующую строку:

source /ifs/apps/parmonc/bin/parmoncvars.sh

Тем самым объявляются три переменные окружения

$PRMCBIN, $PRMCLIB, $PRMCINC.

Эти переменные используются при компиляции и сборке приложений с помощью *PARMONC*, а также при запуске команд. Для компиляции и сборки пользовательских программ с использованием библиотеки *PARMONC* следует использовать следующие команду (приведен пример для программы на C):

mpiicc -o test -LPRMCLIB −lPRMCINC test.c -lparmonc

Здесь *test* - имя исполняемого файла, *test.c* - пользовательская программа.

5. Проблемы масштабирования распределенного статистического моделирования.

Далее приводятся результаты, полученные в рамках реализации федеральной целевой программы "Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2013 годы", государственный контракт 07.514.11.4016.

Поставим вопрос о возможности масштабирования распределенного статистического моделирования на большое число (сотни тысяч и даже миллионы) вычислительных ядер. Актуальность такой постановки обосновывается вычислительной эффективностью алгоритмов распределенного статистического моделирования в свете ожидаемого появления к концу десятилетия суперкомпьютеров экзафлопсного уровня производительности.

Мы имеем в виду задачи статистического моделирования, требующие моделирования экстремально большого количества независимых реализаций. С использованием мультиагентной системы *AGNES* [12] производилось имитационное моделирование работы экзафлопсного суперкомпьютера, загруженного такого рода задачами.

Как известно, теоретическое ускорение при распараллеливании для методов статистического моделирования практически "идеальное", что подтверждается численными расчетами при числе вычислительных ядер порядка нескольких тысяч [5, 11]. Тем не менее, при числе ядер порядка сотен тысяч или нескольких миллионов вопросы организации счета требуют серьезного исследования, поскольку при этом возникают проблемы с большой загрузкой ядер, которые периодически собирают расчетные данные с других ядер.

Целесообразно осуществлять периодическую пересылку результатов промежуточного осреднения реализаций, независимо полученных на загруженных ядрах (ядрах-"вычислителях"), на выделенные ядра (ядра-"сборщики"), объединенные в многоуровневую структуру. Ядра-"сборщики" будут периодически получать переданные им данные и осреднять их, передавая затем результаты на ядро (с номером 0), соответствующее вершине многоуровневой структуры. Будем называть такое ядро главным ядром-"сборщиком"; в числе его задач - сохранение осредненных данных на диск. Рассчитанные на главном ядре-"сборщике" осредненные значения будут соответствовать выборке, полученной совокупно на всех ядрах-"вычислителях". Распределенное статистическое моделирование на разных вычислительных ядрах-"вычислителях" производится в асинхронном режиме. Отправка и получение результатов статистического моделирования также осуществляется в асинхронном режиме.

Далее приводятся некоторые результаты по оценке масштабируемости, полученные путем решения конкретной задачи динамики разреженного газа по методу прямого статистического моделирования, связанному с моделированием реализаций ансамбля тестовых частиц. На кластере НКС-30Т СССЦ с использованием библиотеки *PARMONC* был произведен ряд расчетов для общего числа ядер от 48 до 968. Реальные затраты машинного времени на независимое моделирование реализаций на ядрах-«вычислителях» и обмен данными (выборочными средними) с главным ядром-«сборщиком» были использованы для калибровки имитационной модели в *AGNES*. По результатам расчетов был сделан вывод, что требуемый уровень относительной статистической погрешности в 0.1% достигается при объеме выборки L, равном 240 000. Среднее время моделирования одной реализации составило 12 сек. Для ядер-«вычислителей» обмен данными с главным ядром-«сборщиком» происходил после каждой смоделированной на них реализации. В свою очередь, главное ядро-«сборщик» получало данные с ядер-«вычислителей» после каждой смоделированной на нем реализации.

При имитационном моделировании с использованием *AGNES* предполагалось, что архитектура экзафлопсного суперкомпьютера не отличается от архитектуры кластера НКС-30Т. Рассматривались два варианта организации обмена данными с главным ядром-«сборщиком»: одноуровневый и двухуровневый. В двухуровневом варианте ядра-«вычислители» были поделены на N равных частей (L = 10, 20, 100), для каждой из которых данные с ядер-«вычислителей» сначала отправлялись на свое выделенное промежуточное ядро-«сборщик». В свою очередь, N промежуточных ядер-«сборщиков» отправляли данные на главное ядро-«сборщик». В одноуровневом варианте (будем считать, что число промежуточных ядер-«сборщиков» равно нулю: N = 0) данные с ядер-«вычислителей» непосредственно отправлялись на главное ядро-«сборщик» (см. рис. 4).

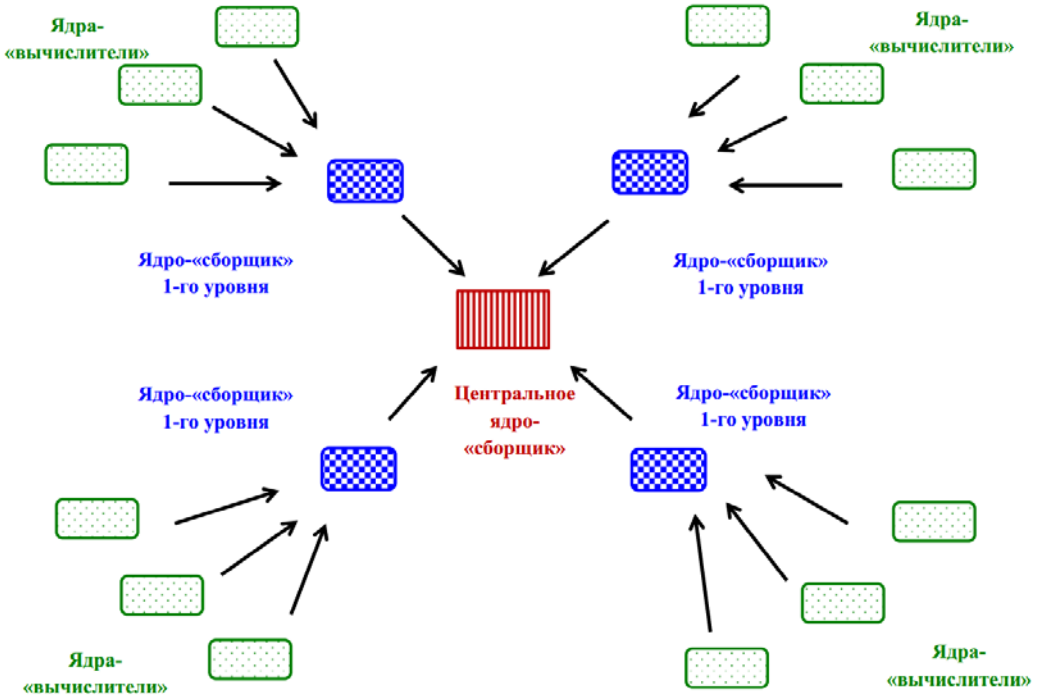

Рис. 4. Двухуровневая система обмена данными при имитационном моделировании.

Под ускорением мы понимаем выражение

$$S_L(M) = T_L(M_{min}) / T_L(M)$$

$T_L(M)$ - машинное время на 0-м ядре-«сборщике», затраченное им на получение данных с других ядер, осреднение и сохранение данных на диск (общее число ядер равно M), M_{min} - минимальное количество использованных при эксперименте процессоров. Отметим, что $T_L(M)$ является убывающей функцией от M.

Теоретическая оценка ускорения для распределенного статистического моделирования (в предположении о пренебрежимо малом времени на обмен данными) следующая:

$$S_L(M) = M / M_{min}$$

Таким образом, теоретически для распределенного статистического моделирование ускорение - «идеальное».

Под масштабированием алгоритмов распределенного статистического моделирования понимается следующее. При фиксированной точности расчетов, т.е. при фиксированном числе реализаций, независимое моделирование реализаций запускается на большем числе ядер. Такой подход позволяет сравнивать время расчетов (и ускорение) на центральном ядре-«сборщике» для разного числа используемых ядер.

На рис. 5-8 приводится сравнение ускорения от распараллеливания для двух вариантов организации обмена данными (под ускорением, как обычно, понимается отношение времени расчетов для одного ядра и времени расчетов для M ядер). В этом случае время расчетов измеряется на главном ядре-«сборщике», а именно, измеряется машинное время, затраченное им на получение данных с других ядер (общее число ядер равно M), осреднение и сохранение данных на диск, причем общее число смоделированных реализаций составляет L. В имитационной модели число ядер M изменялось от 48 до 500000.

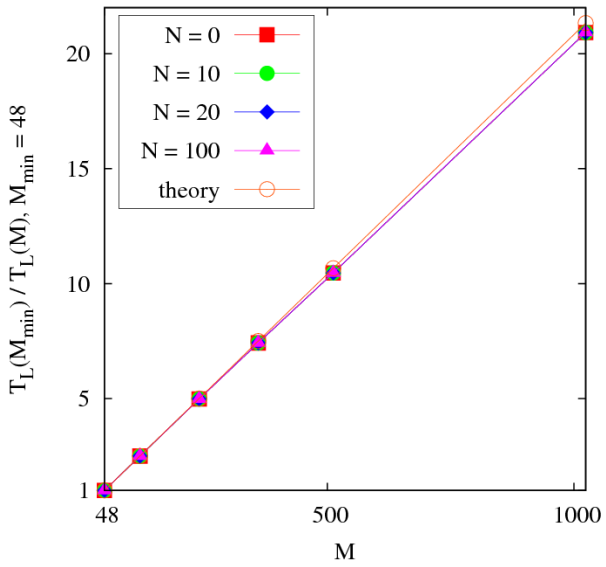

Рис.5. Сравнение ускорения распределенного статистического моделирования для разных вариантов организации обмена данными для числа ядер M до 1000.

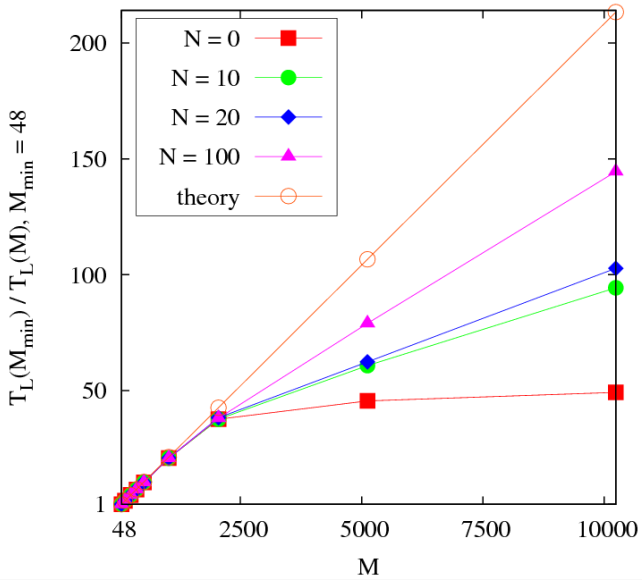

Рис. 6. Сравнение ускорения распределенного статистического моделирования для разных вариантов организации обмена данными для числа ядер M до 10 000.

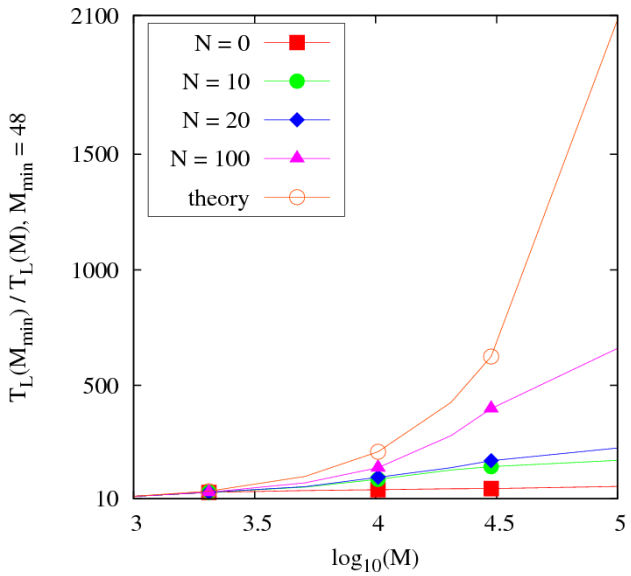

Рис. 7. Сравнение ускорения распределенного статистического моделирования для разных вариантов организации обмена данными для числа ядер M до 100 000 (горизонтальная ось – в логарифмическом масштабе).

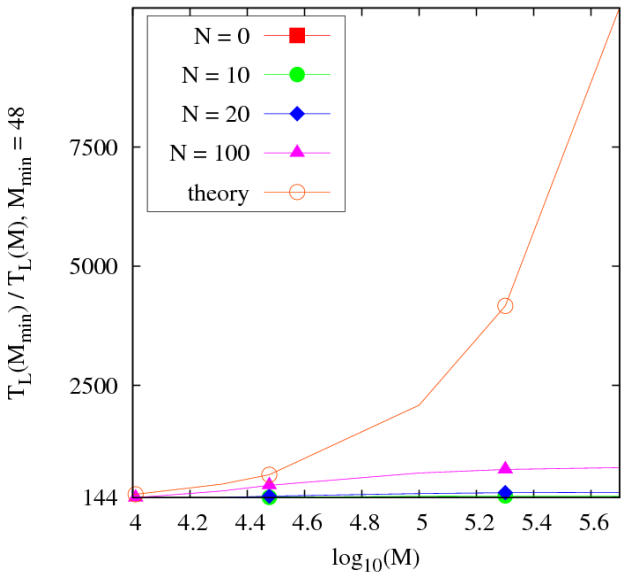

Рис. 8. Сравнение ускорения распределенного статистического моделирования для разных вариантов организации обмена данными для числа ядер M до 500 000 (горизонтальная ось – в логарифмическом масштабе).

По результатам проведенного имитационного моделирования можно сделать следующие выводы. Показано, что даже при явном распараллеливании алгоритма прямого статистического моделирования на большое количество ядер не происходит ожидаемого ускорения, близкого к линейному закону. Это связанно с тем, что при числе ядер порядка сотен тысяч или нескольких миллионов вопросы организации счета требуют серьезного исследования, поскольку при этом возникают проблемы с большой загрузкой ядер-"сборщиков", которые периодически собирают статистику с ядер-"вычислителей". Следо-

вательно, при масштабировании необходима корректировка алгоритма вычислений, в частности, как было показано выше, увеличение количества ядер-"сборщиков".

СПИСОК ЛИТЕРАТУРЫ

1. *Глинский Б.М., Родионов А.С., Марченко М.А., Подкорытов Д.И., Винс Д.В.* Агентно-ориентированный подход к имитационному моделированию суперЭВМ экзафлопсной производительности в приложении к распределенному статистическому моделированию // Вестник ЮУрГУ, 2012. № 18 (277), Вып. 12., с. 93-106.
2. *Glassman I., Yetter R.* Combustion, 4 ed. Academic Press, 2008. 800 p.
3. *Аккерман А.Ф.* Моделирование траекторий заряженных частиц в веществе. М., 1991, 200 с.
4. *Михайлов Г.А., Войтишек А.В.* Численное статистическое моделирование. Методы Монте-Карло. М.: Академия, 2006. 368 с.
5. *Марченко М.А., Михайлов Г.А.* Распределенные вычисления по методу Монте-Карло // Автоматика и телемеханика. 2007. Вып. 5. С. 157–170.
6. *Marchenko M.A.* Majorant frequency principle for an approximate solution of a nonlinear spatially inhomogeneous coagulation equation by the Monte Carlo method // Russ. J. Numer. Anal. Math. Modelling. 2008. V. 21. № 3. P. 199-218.
7. *Marchenko M.A. Mikhailov G.A.* Parallel realization of statistical simulation and random number generators. // Russ. J. Numer. Anal. Math. Modelling. 2002. V. 17. № 1. P. 113-124.
8. *Marchenko M.A.* Parallel Pseudorandom Number Generator for Large-scale Monte Carlo Simulations // LNCS. 2007. V. 4671. P. 276-282.
9. Страница библиотеки PARMONC на сайте ССКЦ КП СО РАН [Электронный ресурс]. – Режим доступа: http://www2.sscc.ru/SORAN-INTEL/paper/2011/parmonc.htm
10. *Marchenko M.* PARMONC - A Software Library for Massively Parallel Stochastic Simulation // LNCS. 2011. V. 6873. P. 302-315.
11. Описание кластера НКС-30Т на сайте ССКЦ КП СО РАН [Электронный ресурс]. – Режим доступа: http://www2.sscc.ru/HKC-30T/HKC-30T.htm
12. *D. Podkorytov, A. Rodionov, H. Choo,* Agent-based simulation system AGNES (AGent NEtwork Simulator) for networks modeling // Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, ICUIMC'12, 2012