Метод декомпозиции области для суперкомпьютерного моделирования гравитирующих систем*

Н.В. Снытников, В.А.Вшивков

Институт Вычислительной Математики и Математической Геофизики СО РАН

Представлен новый параллельный алгоритм для суперкомпьютерного моделирования бесстолкновительных гравитирующих систем. Он комбинирует метод частиц в ячейках (PIC) для решения уравнения Власова и метод параллельного вычисления гравитационного потенциала изолированных систем на основе метода свертки.

Алгоритм использует специальный способ декомпозиции области, динамически назначающий процессоры подобластям и обеспечивающий равномерную балансировку загрузки. Он учитывает физические особенности задач моделирования нестационарных вращающихся дисков (двумерных и трехмерных). В этом случае PIC-частицы, описывающие динамику бесстолкновительного вещества, могут перемещаться по всей вычислительной области и, таким образом, многократно переходить между подобластями и соответствующим им процессорам. Параллельный алгоритм может быть использован на суперкомпьютерах с разными типами архитектуры: традиционными (CPU), и гибридными (CPU с видеокартами NVIDIA GPU и Intel Xeon Phi).

В статье подробно описана теоретическая часть алгоритма. Подробные результаты численных экспериментов будут представлены в отдельной публикации.

1. Введение

Суперкомпьютерное моделирование динамики гравитирующих систем (таких как галактики или околозвездные диски [1]) является одной из наиболее трудных и актуальных проблем в современной вычислительной астрофизике. Хотя однопроцессорные численные методы были созданы и успешно используются уже более 30 лет [2–6], разработка соответствующих им параллельных алгоритмов требует значительных модификаций. В первую очередь это связано постоянным развитием аппаратного обеспечения суперкомпьютеров — увеличивается общее число процессоров, изменяется скорость межпроцессорных коммуникаций, появляются массивно параллельные архитектуры (типа GPU или Phi) — поэтому параллельные алгоритмы и программный код, созданный для архитектур предыдущего поколения, устаревает и нуждается в постоянной переработке. Другой особенностью является разнообразие моделируемых астрофизических процессов [7–9], которые должны быть тесно сопряжены друг с другом. Здесь возникает еще одна трудность в использовании уже созданных программные комплексов и кодов — зачастую они предназначены для решения определенных типов задач (например, задач космологии [10–12]) и не могут быть эффективно использованы в других контекстах.

В данной статье мы предлагаем параллельный алгоритм для моделирования вращающихся бесстолкновительных гравитирующих систем методом частиц в ячейках. Он основан на декомпозиции области для ранее разработанной модели [13]. Алгоритм может быть применен как для модели тонкого диска (когда отсутствует движение вещества в вертикальном направлении), так и для полностью трехмерной модели. Новым в этом подходе является метод динамической балансировки загрузки процессоров при вычислении траектории мо-

*Разработка теоретических основ алгоритма параллельного метода декомпозиции области для метода частиц в ячейках выполнена при поддержке Российского Научного Фонда, грант № 14-11-00485. Программная реализация алгоритмов выполнена при поддержке РФФИ, гранты № 14-01-31088, № 14-01-00392.

дельных частиц, которые в случае вращающихся систем могут многократно переходить между подобластями (и соответствующим им процессорам).

Решаемая система уравнений состоит из уравнения Власова (известного также как бесстолкновительное уравнение Больцмана) для функции распределения вещества $f = f(t, \mathbf{r}, \mathbf{u})$, зависящей от времени t, пространственной координаты $\mathbf{r}$ и вектора скорости $\mathbf{u}$, с заданным начальным значением $f^0(\mathbf{r}, \mathbf{u})$ (в виде некоторого вращающегося диска):

$$\frac{\partial f}{\partial t} + \mathbf{u}\frac{\partial f}{\partial \mathbf{r}} - \nabla\Phi(t, \mathbf{r})\frac{\partial f}{\partial \mathbf{u}} = 0, \quad f(0, \mathbf{r}, \mathbf{u}) = f^0(\mathbf{r}, \mathbf{u})$$

и уравнения Пуассона для гравитационного потенциала $\Phi = \Phi(t, \mathbf{r})$ с краевыми условиями для изолированных систем:

$$\Delta\Phi(t, \mathbf{r}) = 4\pi G\rho(t, \mathbf{r}), \quad \Phi(t, \mathbf{r})|_{|\mathbf{r}|\to\infty} = 0,$$

где G – гравитационная постоянная.

Замыкает систему следующее уравнение для плотности $\rho = \rho(t, \mathbf{r})$, которое связывает ее с функцией распределения f:

$$\rho(t, \mathbf{r}) = \int\limits_{\mathbf{u}} f(t, \mathbf{r}, \mathbf{u}) d\mathbf{u}.$$

Для решения уравнения Власова используется метод частиц в ячейках [2, 3], а для решения уравнения Пуассона — параллельный метод свёртки. Однопроцессорный алгоритм описывается следующими шагами:

1. В исходной вычислительной области вводится равномерная сетка в декартовых координатах. Генерируется начальное распределение модельных частиц (их пространственные координаты и скорости).

2. Решается уравнение Власова с помощью метода частиц. Поскольку PIC-частицы взаимодействуют друг с другом только через гравитационное поле, то их координаты и скорости на следующем шаге могут вычисляться независимо друг от друга. Вместе с тем каждая частица должна для этого «знать» гравитационное поле, вклад в которое вносят все остальные частицы.

3. Вычисляется сеточный гравитационный потенциал с использованием метода свертки (на основе фундаментального решения уравнения Пуассона и быстрого преобразования Фурье).

Соответствующий параллельный алгоритм должен обладать следующими свойствами: (а) подразделять исходную вычислительную область на такие подобласти, что каждая из них могла бы быть обработана одним вычислительным устройством с 1-4 ГБ памяти, и (б) распределять PIC-частицы между процессорами так, чтобы они могли иметь доступ к значениям требуемых сеточных функций в подобластях, свободно двигаться между подобластями, и общее количество частиц, передаваемых между процессорами, было минимальным.

В типичном случае максимальное количество частиц, которое можно поместить в одно вычислительное устройство (CPU, GPU, Intel Phi), составляет 8-16 миллионов, а максимальные количество узлов сетки составляет $256 \times 256 \times 256$ для трехмерных задач и 4096×4096 для двумерных. Это означает, что большие вычислительные области (такие как $1024 \times 1024 \times 1024$) должны подразделяться на меньшие подобласти. В то же самое время PIC частицы могут многократно перелетать из одной области в другую за время одного расчета (например, если мы рассматриваем моделирование дисков на временных масштабах порядка десятков оборотов, где каждая частица двигается по эллиптической

или эпициклической орбите вокруг общего центра масс), что создает проблему пересылок данных на каждом временном шаге.

Эта проблема усугубляется из-за того, что частицы могут быть распределены неравномерно между подобластями. Во вращающемся диске могут возникать гравитационные неустойчивости, которые приводят к появлению разнообразных структур — сгустков вещества или спиральных волн, которые способны передвигаться по всей вычислительной подобласти. Плотность вещества (и количество частиц) в этих структурах может быть на порядки больше фоновых значений. Следовательно, количество частиц в каждой подобласти может так же отличаться на порядки и меняться со временем, особенно в тех случаях, когда сгусток, содержащий большое количество частиц, двигается вокруг центра масс.

Разработанный нами параллельный алгоритм, нацеленный на решение указанных вопросов, основан на следующих предположениях:

- Трудоемкость решения уравнения Власова (т.е. интегрирование траекторий PIC-частиц) существенно выше (от 10 до 100 раз), чем трудоемкость вычисления гравитационного потенциала. Это предположение обосновывается тем, что вычислительная сложность для решения уравнения Пуассона составляет $O(n \log n)$ (где n это общее количество сеточных узлов), в то время как среднее число PIC-частиц на одну ячейку сетки (или один сеточный узел) в типичных расчетах находится в диапазоне $10 \div 100$. Кроме того, количество арифметических операций выполняемых для вычисления координат и скоростей одной частицы больше, чем число операций, выполняемых для одной ячейки. Поэтому возможен алгоритм, когда PIC-частицы обрабатываются с помощью массивно параллельных вычислительных устройств (GPU, Phi), в то время как гравитационный потенциал вычисляется на CPU.

- Поскольку временной шаг, используемый для метода интегрирования траекторий частиц, должен удовлетворять условию Куранта (в противном случае численный метод будет вычислительно неустойчивым), то частица может перелететь за один шаг не далее, чем в соседнюю ячейку. Это означает, что число частиц, которые должны перемещаться между смежными подобластями, намного меньше, чем общее число частиц в смежных подобластях. Это число может быть оценено как $10 \cdot n_y$ для 2D задач и $10 \cdot n_y n_z$ для 3D задач, где коэффициент 10 соответствует оценочному числу перелетающих частиц, а n_y, n_z — это число всех граничных узлов между подобластями.

Общая схема параллельного алгоритма представлена ниже как Алгоритм 1. Он разбит на несколько отдельных программных процедур, которые сформулированы в виде Алгоритмов 2-6.

2. Метод декомпозиции области

Алгоритм 1. Общая схема метода декомпозиции области для параллельного решения.

Исходная вычислительная область (2D или 3D) равномерно подразделяется на K подобластей одинакового размера $S_1, S_2, .., S_K$ в направлении X. Группа процессоров G_k назначается каждой подобласти S_k. Количество процессоров в группе G_k равно P_k, $P_k \geq 1$. Общее количество процессоров равно $P = \sum P_k$. Количество узлов (ячеек) в каждой вычислительной подобласти выбирается таким образом, что все сеточные функции (гравитационный потенциал, плотность, сила) должны полностью помещаться в оперативную память одного вычислительного устройства и оставить в памяти достаточно места для хранения массивов с частицами (т.е. их пространственных координат и скоростей). Обычно это число равно $0.5 \div 2$ миллионов узлов (соответствует сетке 128^3 или 1024^2). Количество процессоров в группе G_k выбирается так, чтобы обеспечить приблизительно одинаковое количество частиц на одном процессоре (см. Алгоритм 2). Это означает, что если подобласть S_{k_1} содержит больше частиц, чем подобласть S_{k_2}, то число процессоров P_{k_1} в группе G_{k_1} будет больше

или равно количеству P_{k_2} в группе G_{k_2}. В каждой группе процессоров G_k и соответствующей подобласти S_k имеется главный процессор g_k^0, который не может быть переназначен любой другой подобласти во время вычислительного эксперимента (даже в том случае, когда подобласть S_k не содержит частиц).

1. В нулевой момент времени ($t = 0$):
   (a) для кажой подобласти S_k вычисляется общее количество частиц, соответствующее числу процессоров в группе G_k, и количество частиц, назначенное каждому из процессоров (см. Алгоритм 2),
   (b) на каждом процессоре выделяется память для массивов координат и скоростей частиц,
   (c) выполняется генерация начальных координат и скоростей PIC частиц в области (на каждом процессоре) в соответствии с заданной пользователем функцией распределения.
2. Вычисляется сеточная функция плотности для PIC частиц (см. Алгоритм 3).
3. Вычисляется гравитационный потенциал с помощью параллельного метода свертки (см. Алгоритм 6).
4. Для каждого процессора вычисляются новые координаты частиц для следующего шага (см. Алгоритм 4).
5. Вычисляется количество частиц, которое должно находиться в каждой подобласти S_k на следующем шаге. Вычисляется количество процессоров, которое должно быть назначено каждой подобласти (см. Алгоритм 2). Перераспределяются частицы между процессорными группами (см Алгоритм 5).
6. Инкрементируется временной шаг и выполняется переход на Шаг 2.

Алгоритм 2. Вычисление числа процессоров в группе.

Предположим, что каждая подобласть S_k содержит N_k частиц, и общее число частиц $N = \sum N_k$. Мы должны вычислить, сколько процессоров должно быть назначено одной подобласти для обработки N_k частиц и при этом обеспечить приблизительно одинаковую загрузку всех процессоров. Каждая процессорная группа содержит по крайней мере один (главный) процессор g_k^0. На каждом временном шаге мы должны вычислить N_{max} — максимальное число частиц на процессоре. Если процессор содержит в точности N_{max} частиц, то общие вычислительные расходы на этом временном шаге будут определяться данным процессором. Таким образом нашей задачей является минимизация параметра N_{max}. Для этого используется метод бисекции:

1. Вычисляем значения $N_{low} = N/P$ и $N_{upp} = 2 \cdot N/P$. Значение N_{low} меньше, чем желаемое N_{max}, а значение N_{upp} больше, чем оптимальное N_{max}. Присваиваем $N_{max} := (N_{low} + N_{upp})/2$.
2. Вычисляем значения $P_k = (N_k/N_{max}) + 1$. Вычисляем значения $P^* = \sum P_k$. Сравниваем P^* с P:
   - Если $P^* < P$, то $N_{upp} := N_{max}$, выполняется переход на Шаг 3.
   - Если $P^* > P$, то $N_{low} := N_{max}$, выполняется переход на Шаг 3.
   - Если $P^* = P$, то $N_{upp} := N_{max}$, и проверяем: если $N_{upp} - N_{low} < N_{eps}$ (где $N_{eps} \ll N_{max}$ и определяется пользователем), то переход на завершение, иначе выполняется переход на 3.

3. Присваиваем $N_{max} := (N_{low} + N_{upp})/2$ и выполняем переход на Шаг 2.

Вычисление плотности и интегрирование траекторий частиц выполняется с помощью стандартного варианта метода частиц в ячейках [3]. Вместе с тем, для эффективной реализации параллельного алгоритма необходимо иметь подходящие структуры данных, комбинирующих частицы и ячейки. Алгоритмы 3-4 определяют основные свойства таких структур данных.

Алгоритм 3. Вычисление плотности внутри подобласти.

- Частицы сортируются внутри каждой подобласти. Ключом сортировки является порядковый номер частицы (порядок устанавливается сначала по координате X, затем по Y, и далее по Z). Эта разновидность сортировочного алгоритма имеет линейную сложность и может быть сделана «на лету» во время интегрирования траекторий частиц (см. Алгоритм 4).

- Далее частицы перераспределяются между процессорами этой группы в соответствии с их порядковыми номерами в отсортированном массиве.

- Подобная процедура сортировки предоставляет возможность эффективного использования кэш-памяти процессора при вычислении координат PIC-частиц.

- Сеточная функция плотности вычисляется с использованием ядра PIC (мультилинейной интерполяции в узлы ячейки). И после этого функция плотности собирается в единый массив на процессоре g_k^0.

Алгоритм 4. Вычисление координат частиц (интегрирование их траекторий).

- Реализована специальная структура данных: каждая ячейка имеет ссылку на массив частиц, которые на данном шаге находятся в данной ячейке.

- Для каждой ячейки интегрируются траектории группы частиц, локализованных в ней.

- При перемещении частиц из одной ячейки в другую, выполняется передача частиц из массива частиц связанных с одной ячейкой, в массив частиц, связанных с другой ячейкой.

- Частицы, которые переместились в ячейки, расположенные за пределами данной подобласти (или за пределами ячеек, назначенных данному процессору), помечаются как «внешние» частицы и позже перемещаются в соответствующую подобласть (см. Алгоритм 5).

Алгоритм 5. Перераспределение частиц между подобластями после вычисления новых координат частиц.

1. После применения Алгоритма 4 на каждом процессоре могут находиться три типа частиц: (а) частицы, которые остались внутри данной подобласти, (б) частицы, которые должны переместиться в левую смежную подобласть, и (в) частицы, которые должны переместиться в правую смежную подобласть. Общее количество перемещаемых между подобластями частиц невелико из-за необходимости выполнения условия Куранта: оно не может быть больше, чем число частиц, которые содержались в граничных ячейках подобласти. На практике это число существенно меньше и составляет менее 0.1% от общего количества частиц.

2. Вычисляем новое число частиц для каждой из подобластей (на основе Алгоритма 4).

3. Вычисляем число процессоров для каждой подобласти (см. Алгоритм 2).

4. Определяем те процессоры в каждой группе, которые должны быть переназначены на соседние подобласти (то есть переданы другим группам). Передаем с них те частицы, которые должны остаться в данной подобласти.

5. Передаем частицы, которые перешли из данной подобласти в соседние подобласти.

3. Параллельный алгоритм для вычисления потенциала методом свёртки

Для вычисления гравитационного потенциала был реализован метод свёртки, предложенный Хокни [2]. Его суть заключается в том, что вместо задачи Дирихле для уравнения Пуассона в бесконечной области:

$$\begin{aligned} \Delta\Phi(\mathbf{x}) &= \rho(\mathbf{x}), \\ \Phi(\mathbf{x})|_{\mathbf{x}\to\infty} &= 0, \end{aligned} \tag{1}$$

выполняется эффективное вычисление интеграла, представляющего фундаментальное решение уравнения Пуассона:

$$\Phi(\mathbf{x_0}) = -\int \frac{\rho(\mathbf{x})d\mathbf{x}}{|\mathbf{x_0}-\mathbf{x}|}. \tag{2}$$

В дискретном случае в декартовой системе координат на равномерной сетке с числом узлов $N_x \times N_y \times N_z$ и сеточными шагами h_x, h_y, h_z решение будет записываться в следующем виде:

$$\Phi(x_0, y_0, z_0) = -\sum_{i=1}^{N_x-1}\sum_{j=1}^{N_y-1}\sum_{k=1}^{N_z-1} \frac{q_{i,j,k}}{\sqrt{(x_i-x_0)^2+(y_i-y_0)^2+(z_i-z_0)^2}}, \tag{3}$$

где $q_{i,j,k} = \rho_{i,j,k} \cdot (h_x h_y h_z)$ — значения зарядов (масс), находящихся в узлах сетки.

Нетрудно видеть, что прямое вычисление этой суммы для всех x_0, y_0, необходимое для восстановления сеточной функции гравитационного потенциала потребует $O(N_x^2 N_y^2 N_z^2)$ операций. Однако трудоемкость вычислений можно существенно сократить, воспользовавшись теоремой о свёртке [14] и ее дискретным аналогом. Запишем (2) в виде:

$$\Phi(\mathbf{x_0}) = -\int \rho(\mathbf{x})K(\mathbf{x}-\mathbf{x_0})d\mathbf{x} = -\rho * K, \tag{4}$$

где

$$K(\mathbf{x}-\mathbf{x_0}) = \frac{1}{|\mathbf{x}-\mathbf{x_0}|}$$

и $\rho * K$ обозначает свёртку.

Если интеграл в (4) является абсолютно интегрируемым и существуют следующие интегралы (т.е. существуют ограничивающие константы C_1 и C_2):

$$\begin{aligned} \int \rho(\mathbf{x})d\mathbf{x} < C_1 < \infty, \\ \int K(\mathbf{x})d\mathbf{x} < C_1 < \infty, \end{aligned} \tag{5}$$

то:

$$\begin{aligned} FT[\Phi](\mathbf{k}) &= -FT[\rho](\mathbf{k}) \cdot FT[K](\mathbf{k}), \\ \Phi &= -FT^{-1}\left[FT[\rho] \cdot FT[K]\right] \end{aligned} \tag{6}$$

где $FT[\ldots]$ обозначает преобразование Фурье.

С практической точки зрения это означает, что, воспользовавшись дискретным аналогом теоремы о свёртки и алгоритмом быстрого преобразования Фурье, можно вычислить сумму (3) за $O(N_xN_yN_z(log_2N_x+log_2N_y+log_2N_z))$ операций, что намного быстрее прямого способа вычисления фундаментального решения для уравнения Пуассона и соответствует трудоемкости решения задачи Дирихле для уравнения Пуассона методом разделения переменных [15].

Для того, чтобы применить дискретное преобразование Фурье необходимо устранить неопределенность интеграла (5) в точке $\mathbf{x} = \mathbf{x_0}$, где он расходится, и определить функции K и ρ так, чтобы они стали периодическими. Первая из этих проблем решается с помощью обрезания потенциала на близких расстояниях. Мы задавали сеточную функцию K следующим образом:

$$K(x,y,z) = \begin{cases} \frac{1}{0.5\min(hx,hy,hz)}, & \sqrt{x^2+y^2+z^2} = 0, \\ \frac{1}{\sqrt{x^2+y^2+z^2}}, & \sqrt{x^2+y^2+z^2} > 0. \end{cases}$$

Вторая проблема решается с помощью введения фиктивных дополнительных подобластей, дублирующих область решения по каждому направлению в два раза, и доопределения в них функции K так, чтобы она стала периодической во всей новой области, а функция ρ равной нулю. Аккуратное доказательство корректности метода и того факта, что полученное таким способом решение совпадает с прямым вычислением (3), приведено в статье [16].

Производительность представленного алгоритма свёртки фактически определяется производительностью быстрого преобразования Фурье (в нашей реализации метода свёртки мы использовали FFTW [17]).

Параллельная реализация данного алгоритма заключается в разбиении на подобласти и применени метода транспозиции данных (стандартного подхода для многомерного преобразования Фурье [18]).

Алгоритм 6. Параллельный алгоритм для вычисления гравитационного потенциала.

1. Вычислительная область в 2D или 3D подразделяется на подобласти в направлении X.
2. Применяется прямое быстрое преобразование Фурье (БПФ) к сеточным функциям плотности ядра потенциала в направлении Y и в направлении Z.
3. Выполняется транспозиция «слоев» из направления Y в направление X.
4. Применяется прямое БПФ в направлении X. Перемножается образ сеточной функции ядра на образ функции плотности. Применяется обратное БПФ в направлении X к полученному результату.
5. Выполняется обратная транспозиция «слоев» из направления X в направление Y.
6. Применяется обратное БПФ в направлении Z и обратное БПФ в направлении Y.

Для данного алгоритма единственными межпроцессорными коммуникациями являются прямая и обратная транспозиция данных (соответствующая процедуре MPI Alltoall). Численные эксперименты показали, что для сеток среднего размера (до 1024^3 узлов) и числом процессоров до 1024 коммуникационное время составляет около 50% (не более 1-3 секунд).

4. Предварительные тестовые результаты

Численные тестовые эксперименты проводились на суперкомпьютерах Сибирского суперкомпьютерного центра, Межведомственного суперкомпьютерного центра и суперкомпьютера «Ломоносов» в МГУ. В качестве начального распределения $f^0(\mathbf{r},\mathbf{u})$ задавался диск

Маклорена с твердотельным вращением и разными дисперсиями скоростей частиц. В качестве максимальных технических параметров для квазитрехмерной модели (с отсутствием вертикального движения вещества) задавалась сетка 16384×16384 и 5 миллиардов частиц (512 процессоров), а для полностью трехмерной модели — $1024 \times 1024 \times 1024$ и 10 миллиардов частиц (1024 процессора). Для тестовых задач на развитие гравитационной неустойчивости осесимметричного типа (варьирование начальных дисперсий) общий объем коммуникационных расходов составлял не более 30% с абсолютным временем счета порядка 10 секунд на один шаг по времени.

Подробные результаты численных экспериментов находятся в стадии подготовки и будут представлены в отдельной публикации. В дальнейшем разработанный параллельный алгоритм и его программная реализация будет использоваться для суперкомпьютерного моделирования динамики вращающихся галактик и пылевой компоненты в протопланетных дисках.

Литература

1. Снытников В.Н., Вшивков В.А., Кукшева Э.А., Неупокоев Е.В., Никитин С.А., Снытников А.В. Трехмерное численное моделирование нестационарной гравитирующей системы многих тел с газом // Письма в астрономический журнал. 2004. 30, N.2. 146-160.

2. Hockney, R.W. and Eastwood, J.W. Computer Simulation Using Particles. // McGraw-Hill. New York. 1981.

3. Berezin Yu.A. Vshivkov V.A. Particle-in-cell method in the plasma dynamics // Nauka. 1980.

4. J.E. Barnes and P. Hut. A Hierarchical O(NlogN) Force-Calculation Algorithm // Nature Vol. 324, pp.446-449, 1986.

5. R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics - Theory and application to non-spherical stars // Monthly Notices of the Royal Astronomical Society, vol. 181, p. 375-389. 1977.

6. P. Colella, P.R. Woodward. The Piecewise Parabolic Method (PPM) for gas-dynamical simulations // Journal of Computational Physics. Volume 54, Issue 1, P. 174-201. 1984

7. A.Dubeya, K. Antypasb, M.K. Ganapathyc, et al. Extensible component-based architecture for FLASH, a massively parallel, multiphysics simulation code // Parallel Computing. Volume 35. 2009. P.512-522.

8. Springel V., Yoshida N., White S.D.M.: GADGET: a code for collisionless and gasdynamical cosmological simulation. New Astronomy. 6, pp. 79-117 (2001)

9. F.R. Pearce, H.M.P. Couchman. Hydra: a parallel adaptive grid code // New Astronomy. Volume 2, Issue 5, P. 411-427. 1997.

10. V. Springel et al. Simulations of the formation, evolution and clustering of galaxies and quasars // Nature, V. 435, p. 629. 2005,

11. A.A. Klypin et al. Dark Matter Halos in the Standard Cosmological Model: Results from the Bolshoi Simulation // Astrophysical Journal. V. 740 p. 102. 2011.

12. Y. Feng, T. Di Matteo, R.A.C. Croft, S. Bird, N. Battaglia, S. Wilkins // Monthly Notice of Royal Astronomical Society. 2015 (submitted)

13. N. Snytnikov. Scalable Parallel Algorithm for Solving the Collisionless Boltzmann - Poisson System of Equations // Astronomical Society of the Pacific Conference Series. - 2012. - V. 453. - P. 393.

14. Kolmogorov A.N., Fomin S.V. Elements of the Theory of Functions and Functional Analysis // Graylock Press, Rochester, N.Y., 1961.

15. Самарский А.А. Николаев Е.С. Методы решений сеточных уравнений. М.: Наука. 1978. 592 с.

16. Eastwood J.W., Brownrigg D.R.K. Remarks on the Solution of Poisson's Equation for Isolated Systems // Journal of Computational Physics, Vol. 32, pp.24-38, 1979.

17. M.Frigo and S.G.Johnson. FFTW software // http://www.fftw.org

18. O. Ayala, L.P. Wang. Parallel implementation and scalability analysis of 3D Fast Fourier Transform using 2D domain decomposition // Parallel Computing. 2013. Vol.39. P. 58-77