УДК 004.4'2

Д.В. Лебедев, В.А. Перепелкин

Казахский национальный университет им. аль-Фараби, г. Алматы, Казахстан
E-mail: dan-lebedev@mail.ru

Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск, Россия
E-mail: perepelkin@ssd.sscc.ru

Численное решение одномерной краевой задачи нефть-вода и ее реализация в системе фрагментированного программирования LuNA

Кратко представлена технология фрагментированного программирования и реализующие ее язык и система фрагментированного программирования LuNA, на примере решения одномерной краевой задачи нефть-вода. Технология ориентирована на поддержку разработки параллельных программ, реализующих большие численные модели, и их исполнения на суперкомпьютерах. Система LuNA автоматизирует программирование коммуникаций, распределение ресурсов, управление памятью, синхронизацию доступа к общим данным, динамическую балансировку нагрузки на вычислительные узлы и т.п.
Ключевые слова: Фрагментированное программирование, LuNA, численное решение, MPI, параллельная программа, прогонка.

D. Lebedev, V. Perepelkin

Fragmented program numerical solution of one-dimensional boundary value problem in the oil-water system LuNA

We introduce the application of the fragmented programming technology and the LuNA language and fragmented programming system for parallel implementation of the [1d краевая нефть-вода] problem. The technology is oriented to support the implementation of large-scale numerical models for supercomputers. The LuNA system automates programming of communications, resources distribution, memory management, synchronizations, dynamic load balancing etc.
Key words: Fragmented programming, LuNA, numerical solution, MPI, Parallel program.

Д.В. Лебедев, В.А. Перепелкин

.

Түйін сөздер: MPI, LuNA.

Введение

При параллельной реализации больших численных моделей на суперкомпьютерах возникают необходимость решать проблемы из области системного параллельного программирования. К таким проблемам относятся декомпозиция данных и вычислений на части и распределение этих частей по различным узлам мультикомпьютера, обеспечение и синхронизация доступа к распределённым данным, планирование вычислений,

обеспечение динамической балансировки нагрузки на процессоры, и т.д. [1] Решение этих проблем важно автоматизировать, т.к. это повышает уровень программирования, снижает требования к квалификации программиста, обычно повышает качество параллельной программы, увеличивает скорость и снижает трудоёмкость разработки параллельной программы, снижает число возникающих ошибок. Особенно это существенно при программировании на неоднородных вычислителях, так как приходится учитывать соответствующую специфику - разные объемы оперативной и кэш-памятей узлов, разную производительность вычислительных узлов, разные характеристики коммуникационных сетей - топологию, пропускную способность, латентность и т.п.

Подобная автоматизация - сложная задача, которую целесообразно решать в ограниченной предметной области и для ограниченного класса вычислителей. Технология фрагментированного программирования и поддерживающая её система LuNA рассчитаны на автоматизацию параллельной реализации итерационных численных алгоритмов на регулярных сетках на неоднородных суперкомпьютерах, получающихся в результате объединения нескольких кластеров в один, например, с помощью коммуникационной среды NumGRID [3].

Система LuNA

Система параллельного программирования LuNA разрабатывается в ИВМиМГ СО РАН. Она предназначена для эффективной реализации больших численных моделей на мультикомпьютерах с распределенной памятью (в первую очередь, вычислительных кластерах). Тут и далее эффективность понимается с точки зрения расхода памяти и затрачиваемого времени на выполнение программы. Теоретический базис системы LuNA - теория структурного синтеза параллельных программ на вычислительных моделях [2].

Основное преимущество, которое даёт использование системы LuNA - это экономия времени программирования больших численных моделей. Эта экономия достигается за счёт автоматизации программирования многих трудоёмких системных компонентов параллельной программы. В частности, автоматизируется программирование коммуникаций, распределения ресурсов, управления памятью, синхронизации доступа к общим данным, динамической балансировки нагрузки на вычислительные узлы и т.п.

Собственно программирование осуществляется на двух языках программирования - LuNA и C++. Схема прикладного алгоритма описывается на языке LuNA (Language for Numerical Algorithms), который разрабатывается в рамках того же проекта, что и сама система. Этот язык позволяет описывать структурную схему алгоритма как двудольный ориентированный граф, вершинами которого являются фрагменты данных и фрагменты вычислений. Фрагменты данных соответствуют агрегированным переменным алгоритма, это могут быть, например, блоки сеток, подматрицы, и т.п. Фрагменты вычислений соответствуют операциям алгоритма, они задают, как значения одних фрагментов данных могут быть вычислены из других фрагментов данных. Дуги графа определяют для каждого фрагмента вычислений, какие из фрагментов данных являются для него входными, а какие - выходными.

На языке C++ описываются последовательные процедуры без побочных эффектов, которые реализуют фрагменты вычислений. Такое представление прикладного алгоритма будем называть фрагментированной программой. Использование распространенного

последовательного языка программирования для реализации фрагментов вычислений выгодно, так как позволяет основную массу вычислений, сосредоточенную внутри фрагментов вычислений, реализовать с высокой эффективностью на базе накопленного опыта, алгоритмов, библиотечных подпрограмм и инструментов, существующих в области последовательного программирования.

Фрагментированная структура алгоритма сохраняется и во время его выполнения, то есть фрагменты данных и вычислений явно пристутствуют на узлах мультикомпьютера как независимые и взаимодействующие процессы. Исполнение алгоритма автоматически осуществляет Система LuNA. Она размещает фрагменты данных в распределенной памяти мультикомпьютера, отслеживает информационные зависимости между фрагментами вычислений и исполняет последние. Исполнение фрагмента вычислений состоит в том, что ему назначается вычислительный узел, в локальную память этого узла передаются все входные фрагменты данных для этого фрагмента вычислений (если они уже не находятся там), и запускается последовательная процедура над этими фрагментами данных. Результатом выполнения процедуры является набор значений выходных фрагментов данных, которые используются системой далее для выполнения следущих фрагментов выичслений.

Такая схема выполнения прикладного алгоритма позволяет системе LuNA распределять и перераспределять фрагменты данных и вычислений по узлам мультикомпьютера и выбирать порядок их выполнения. Это позволяет осуществлять автоматически балансировку нагрузки на вычислительные узлы, управление памятью, планирование вычислений, настройку программы на конфигурацию вычислителя и т.п., снимая соответствующую работу с человека.

Несмотря на принципиальную возможность автоматически решать перечисленные задачи, на практике это не всегда возможно вследствие алгоритмтической сложности этих задач, многие из которых в общей постановке являются переборными, например, задача планирования вычислений. Вследствие этого в системе LuNA имеется возможность задавать рекомендуемый порядок выполнения фрагментов вычислений, распределения фрагментов данных по узлам и т.п. (далее будем называть это поведением фрагментированной программы). Такие средства являются частью языка LuNA и называются рекомендациями. Рекомендации являются частью фрагментированной программы. Рекомендации используются системой как подсказка при решении оптимизационных задач. Такой подход, с одной стороны, позволяет существенно повысить результирующую эффективность исполнения прикладного алгоритма. С другой стороны, пользователю не требуется программировать желаемое поведение программы, он лишь описывает его в декларативной форме. Изменение рекомендаций не влияет на функциональную часть описания прикладного алгоритма, то есть на собственно вычисляемые величины. Это позволяет изменять поведение программы, добиваясь эффективности ее исполнения без опасности нарушить правильность ее работы.

Система LuNA состоит из двух основных компонентов - это компилятор LuNA и исполнительная система LuNA (см. рис. 1). Прикладной программист описывает численный алгоритм в виде фрагментированной программы, которая компилятором преобразуется в представление, исполняемое исполнительной системой. Компилятор выполняет статический анализ программы и генерирует ряд рекомендаций. В настоящее время рекомендации компилятора не особенно существенно повышают эффективность

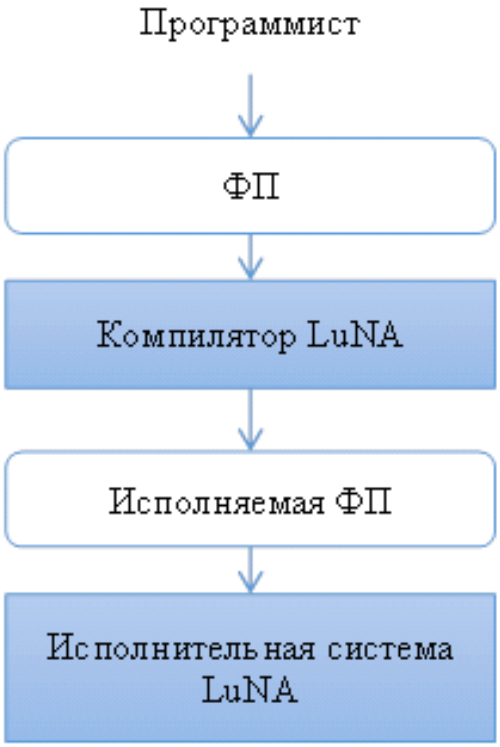

Рисунок 1 – Общая структура системы LuNA (ФП - фрагментированная программа)

исполнения фрагментированной программы, и важность рекомендаций, предоставляемых человеком, существенно выше. Такое положение дел вызвано слабой текущей проработкой алгоритмов компиляции. По мере развития системы LuNA роль автоматически сгенерированных рекомендаций должна повышаться вплоть до полного освобождения человека от этой части работ.

Более подробно с системой LuNA, ее теоретическими и технологическими аспектами можно ознакомиться в [1].

Разработка исходного алгоритма

Будем рассматривать одномерную краевую задачу нефть-вода. Данная задача описывается следующими уравнениями

$$\frac{\partial}{\partial x}\left[K_0\left(1+C_f P_h\left(P_o-1\right)\right)\left(\frac{\partial P_o}{\partial x}-\frac{\gamma_0 L}{P_h}\frac{\partial z}{\partial x}\right)\right]+\frac{\partial}{\partial x}\left[K_w\left(\frac{\partial P_o}{\partial x}-\frac{\partial P_c ow}{\partial x}-\frac{\gamma_w L}{P_H}\frac{\partial z}{\partial x}\right)\right]= C_f P_H\frac{\partial}{\partial \tau}\left(S_o P_o\right)+\frac{\partial}{\partial \tau}\left[\left(1-C_f P_H\right)S_o\right]+\frac{\mu_o}{\mu_w}\frac{\partial S_w}{\partial \tau}+\frac{\mu_o L^2}{K\rho_H P_H}\left(q_w+q_o\right) \quad (1)$$

$$\frac{\partial}{\partial x}\left[K_w\left(\frac{\partial P_w}{\partial x}-\frac{\gamma_w L}{P_H}\frac{\partial z}{\partial x}\right)\right]=\frac{\mu_w}{\mu_o}\frac{\partial S_w}{\partial \tau}+\frac{\mu_o L^2}{K\rho_H P_H}q_w \quad (2)$$

с начальными и граничными условиями

$$\begin{cases} P_o(x,0) = P_o^H(x), P_w(x,0) = P_w^H(x) \\ S_0(x,0) = S_0^H(x), S_w(x,0) = S_w^H(x) \end{cases}, 0 < x < L$$
$$\left.\frac{\partial P_l}{\partial x}\right|_{x=0} = 0, \left.\frac{\partial P_l}{\partial x}\right|_{x=L} = 0, l = (o,w) \tag{3}$$

Для построения разностной схемы построим следующую пространственно-временную сетку:

$$D_{h,\tau k1} = \left\{(x_i,t_k) : x_i = x_{i-1} + h, i = \overline{1,N}, x_0 = 0, x_N = 1, t_k = t_{k-1} + \tau, k = 1,2,\ldots, t_0 = 0\right\}$$

На этой сетке введем следующие обозначения

$$P_o(x_i,t_k) = P_{oi}, P_o(x_i,t_{k-1}) = \overline{P}_{oi}, P_w(x_i,t_k) = P_{wi},$$
$$P_w(x_i,t_{k-1}) = \overline{P}_{wi}, S_o(x_i,t_k) = S_{oi}, S_o(x_i,t_{k-1}) = \overline{S}_{oi}$$
$$S_w(x_i,t_k) = S_{wi}, S_w(x_i,t_{k-1}) = \overline{S}_{wi}, z(x_i) = z_i$$

Для решения системы (1)–(3) применим итерационный метод. Самым эффективным является метод простой итерации Для этого запишем уравнение (1) в конечно-разностном виде

$$a_i^{(r-1)} P_{oi-1}^{(r)} - b_i^{(r-1)} P_o i^{(r)} + c_i^{(r-1)} P_{oi+1}^{(r)} = -f_i^{(r-1)}, i = 1, N-1 \tag{4}$$

Здесь

$$a_i^{(r-1)} = K_{oi-1/2}^{(l-1)} \left(1 + C_f P_H \left(P_{oi-1/2}^{(r-1)} - 1\right)\right) + K_{wi-1/2}^{(l-1)};$$

$$c_i^{(r-1)} = K_{oi+1/2}^{(l-1)} \left(1 + C_f P_H \left(P_{oi+1/2}^{(r-1)} - 1\right)\right) + K_{wi+1/2}^{(l-1)};$$

$$b_i^{(r-1)} = a_i^{(r-1)} + c_i^{(r-1)} + \frac{h^2}{\tau} C_f P_H S_{oi}^{(l-1)};$$

$$f_i^{(r-1)} = \frac{h^2}{\tau}\left[C_f P_H \overline{S}_{oi}\overline{P}_{oi} - (1 - C_f P_H)\left(S_{oi}^{(l-1)} - \overline{S}_{oi}\right) - \frac{\mu_w}{\mu_o}\left(S_{wi}^{(l-1)} - \overline{S}_{wi}\right)\right]$$

$$-K_{oi+1/2}^{(l-1)} \left(1 + C_f P_H \left(P_{oi+1/2}^{(r-1)} - 1\right)\right) \frac{\gamma_o L}{P_H}(z_{i+1} - z_i)$$

$$+K_{oi-1/2}^{(l-1)} \left(1 + C_f P_H \left(P_{oi-1/2}^{(r-1)} - 1\right)\right) \frac{\gamma_o L}{P_H}(z_i - z_{i-1})$$

$$-K_{wi+1/2}^{(l-1)}\left[P_{cowi+1}^{(r-1)} - P_{cowi}^{(r-1)} + \frac{(\gamma_w L}{P_H}(z_{i+1} - z_i)\right]$$

$$+K_{wi-1/2}^{(l-1)}\left[P_{cowi}^{(r-1)} - P_{cowi-1}^{(r-1)} + \frac{\gamma_w L}{P_H}(z_i - z_{i-1})\right] - h^2 \frac{\mu_o L^2}{K \rho_H P_H}(q_w + q_o)$$

Для решения системы (4) на каждом r итерационном слое применим метод прогонки[7,8]. Для завершения процесса необходимо выполнения следующего условия

$$\left\|P_{oi}^{(r)} - P_{oi}^{(r-1)}\right\| \le \varepsilon_1$$

Следующим шагом находим P_w по формуле $P_w = P_o - P_{cow}$ Зная P_w можно найти q_w по формуле

$$q_w = \frac{KK_w}{\mu_w}\left(\frac{dP_w}{dx} - \gamma_w \frac{dz}{dx}\right)$$

которая исходит из закона Дарси Следующим шагом необходимо определить $S_{wi}^{(l)}$, для этого запишем уравнение (2) в конечно-разностном виде и разрешенном относительно S_{wi}. При этом это значение будет являться итерационным, т.к. при его нахождении будут использоваться те значения, которые определены на предыдущих шагах итерации.

$$S_{wi}^{(l)} = \overline{S}_{wi} + \frac{\mu_o \tau}{\mu_w h^2}\left[K_{wi+1/2}^{(l-1)}\left(P_{wi+1} - P_{wi} - \gamma_w \frac{L}{P_H}(z_{i+1} - z_i)\right) - K_{wi-1/2}^{(l-1)}(P_{wi} - P_{wi-1}))\right] - \gamma_w \frac{L}{P_H}(z_i - z_{i-1})\Bigg)\Bigg] - \frac{(\mu_o^2 \tau L^2}{\mu_w K \rho_w P_H} q_w \quad (5)$$

Значения $P_o^{(r)}$, $S_w^{(l)}$, является промежуточными и для уточнения необходимо подставить найденные значения в коэффициенты системы (4) и заново решить эту систему, затем найденные значения подставить в (5). Процесс остановиться при достижении следующего условия

$$\left\|S_w^{(l)} - S_w^{(l-1)}\right\| \le \varepsilon_2$$

Затем находим $S_o = 1 - S_w$, $q_o = 1 - q_w$ За первые приближения $P_o^{(0)}$, $S_w^{(0)}$, берутся значения из предыдущего временного слоя P_o и S_w.

Фрагментация исходного алгоритма и реализация его в системе LuNA

Для того, чтобы исходный алгоритм можно было запрограммировать в системе LuNA, он должен быть фрагментирован, то есть представлен как множество фрагментов данных и вычислений. Для этого произведем разбиение области пространства на m частей, и будем решать исходную задачу внутри каждой подобласти для уравнения (1) методом параллельной прогонки [9], а для уравнения (2) использовать явную схему. Ниже описан один из вариантов метода параллельной прогонки. Пусть каждому фрагменту выделено одинаковое количество узлов сетки $m = M/N$, где M - число узлов сетки, N - число фрагментов, индексация глобальная, если M не делится на N, то для последнего фрагмента число узлов увеличиваем на остаток от деления. Таким образом, на отдельном процессоре с номером j решается лишь часть уравнений системы (1) с номерами от $(j-1)\cdot m + 1$ до $j \cdot m$, где j - номер фрагмента. Обозначим $P_{oj\cdot m}$ через $z_j, j = 0, ..., N$, и представим искомые значения в виде

$$P_{oi}^{(r)} = z_{j-1} \cdot u_i + z_j \cdot v_i + w_i \quad (6)$$

где u, v, w - решения следующих систем уравнений:

$$\begin{cases} a_i^{(r-1)} \cdot u_{i-1} - b_i^{(r-1)} \cdot u_i + c_i^{(r-1)} \cdot u_{i+1} = 0 \\ u_{(j-1)\cdot m} = 1, u_{j\cdot m} = 0 \end{cases}$$
$$\begin{cases} a_i^{(r-1)} \cdot v_{i-1} - b_i^{(r-1)} \cdot v_i + c_i^{(r-1)} \cdot v_{i+1} = 0 \\ u_{(j-1)\cdot m} = 0, u_{j\cdot m} = 1 \end{cases} \quad (7)$$
$$\begin{cases} a_i^{(r-1)} \cdot w_{i-1} - b_i^{(r-1)} \cdot w_i + c_i^{(r-1)} \cdot w_{i+1} = -f_i \\ w_{(j-1)\cdot m} = 0, w_{j\cdot m} = 0 \end{cases}$$

Решения этих трех систем можно найти методом прогонки, причем независимо на каждом из фрагментов. Будем называть эти решения предрешениями, а этот этап решения задачи - этапом нахождения предрешений. В уравнения с номерами $j \cdot m$ подставляем вместо P_o комбинации (6). Таким образом, получаем систему трехточечных уравнений для вычисления z_j , имеющую следующий вид:

$$A_j \cdot z_{j-1} - B_j \cdot z_j + C_j \cdot z_{j+1} = -F_j j = 1, \ldots, N-1$$
$$-B_0 \cdot z_0 + C_0 \cdot z_1 = -F_0, A_N \cdot z_{N-1} - B_N \cdot z_N = -F_N, B_0 = b_0 - c_0 \cdot u_1, C_0 = c_0 \cdot v_1, F_0 = f_0 + c_0 \cdot w_1$$
$$A_j = a_{j\cdot m} \cdot u_{j\cdot m-1}, B_j = b_{j\cdot m} - a_{j\cdot m} \cdot v_{j\cdot m-1} - c_{j\cdot m} \cdot u_{j\cdot m+1}, C_j = c_{j\cdot m} \cdot v_{j\cdot m+1}$$
$$F_j = f_{j\cdot m} + a_{j\cdot m} \cdot w_{j\cdot m-1} + c_{j\cdot m} \cdot w_{j\cdot m+1}, j = 1, \ldots, N-1$$
$$A_N = a_{N\cdot m} \cdot u_{N\cdot m-1}, B_N = b_{N\cdot m} - a_{N\cdot m} \cdot v_{N\cdot m-1}, F_N = f_{N\cdot m} + a_{N\cdot m} \cdot w_{N\cdot m-1} \quad (8)$$

Назовем этот этап решения задачи этапом нахождения решений на границе процессоров. Решение системы (8) и составляет нераспараллеливаемую часть данного алгоритма. Размерность этой системы уравнений равна количеству фрагментов, а не количеству точек задачи. Систему будем решать методом прогонки на одном процессоре. Окончательное решение восстанавливаем по формуле (6) и проверяем условие сходимости $\left\| P_{oi}^{(r)} - P_{oi}^{(r-1)} \right\| \leq \varepsilon_1$ для всех подобластей, если оно не выполняется хотя бы в одной подобласти, то снова решаем уравнения (7) затем (8) и (6). Следующим шагом находим P_w по формуле $P_w = P_o - P_c ow$ Зная P_w можно найти q_w по формуле

$$q_w = \frac{KK_w}{\mu_w} \left(\frac{dP_w}{dx} - \gamma_w \frac{dz}{dx} \right)$$

Затем вычисляем $S_{wi}^{(l)}$ по формуле (5) для каждой подобласти и проверяем условие сходимости $\left\| S_w^{(l)} - S_w^{(l-1)} \right\| \leq \varepsilon_2$ Узким местом данного алгоритма является то, что для вычисления z_j необходимо знать значения $u_{j\cdot m-1}$, $u_{j\cdot m+1}$, $v_{j\cdot m-1}$, $v_{j\cdot m+1}$, $w_{j\cdot m-1}$, $w_{j\cdot m+1}$ которые вычисляются независимо. Также для вычисления A_j, B_j, C_j, F_j необходимо знать $P_{oi}^{(r-1)}$, $\overline{P}_{oi}$, $S_{wi}^{(l-1)}$, $\overline{S}_{wi}$, $\overline{S}_{oi}$. То есть при нахождении z_j, эти значения могут храниться на других узлах. Эти данные должны быть переданы перед началом вычислений фрагмента. Еще одним узким местом является проверка условий сходимости. На основании этого можно сделать вывод, что количество фрагментов не должно быть слишком большим, т.к. много времени будет тратиться на передачу данных.

Тестирование

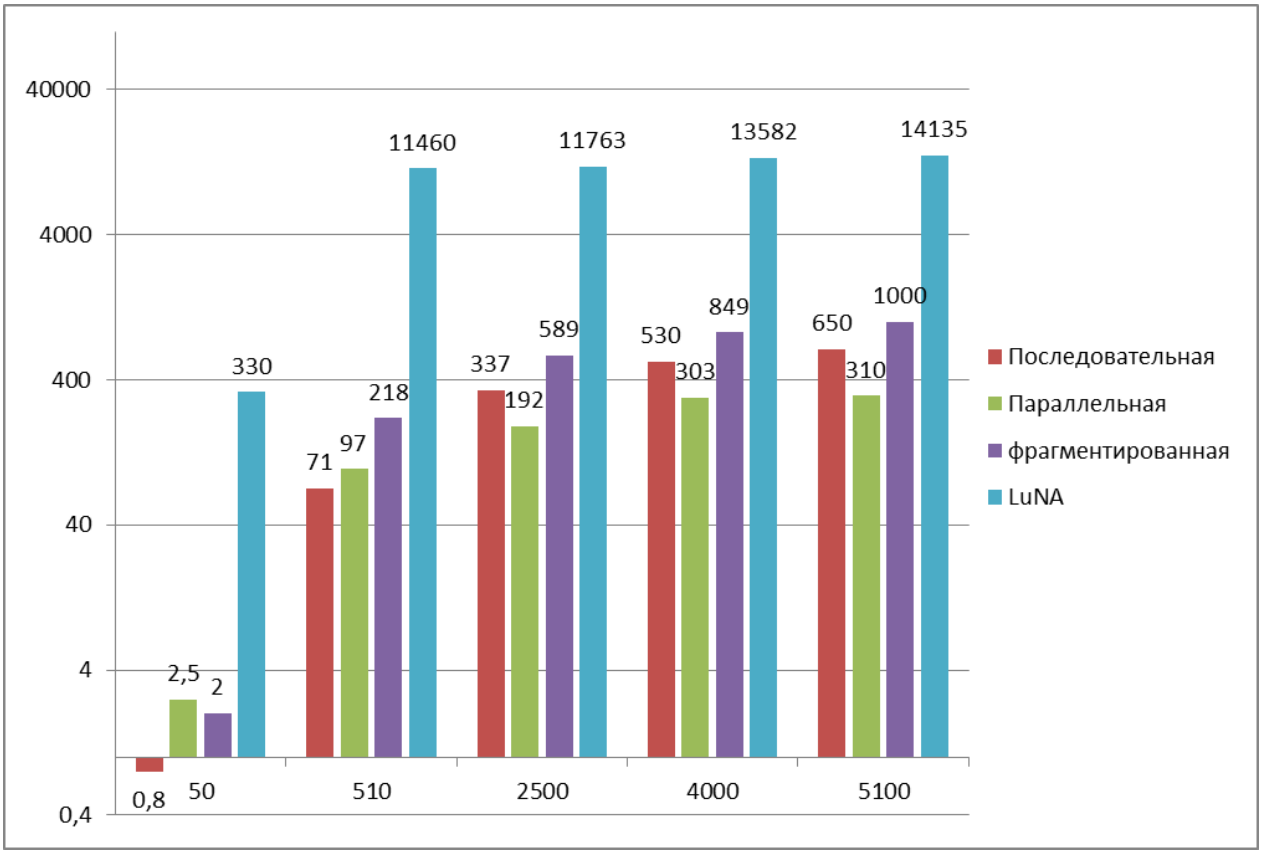

Рисунок 2 – Время расчета (в сек.) в зависимости от числа точек сетки

Расчеты проводились для различного количества точек по оси x. Для сравнения качества реализаций прикладной численный алгоритм был реализован в нескольких вариантах: последовательная программа на языке Java, параллельная программа на языке Java с использованием коммуникационной библиотеки MPJ Express, реализующей стандарт MPI, последовательная фрагментированная программа на языке C++ и параллельная фрагментированная программа на языке LuNA с использованием одноименной системы программирования. Целью тестирования являлось определение эффективности различных реализаций рассматриваемого численного алгоритма. Последовательная и фрагментированная версия запускались на одном ядре (последовательно). Параллельная версия запускалась на 4-ядерном компьютере, а LuNA-версия - на 32 ядрах в распределённой памяти (по одному ядру на вычислительный узел) . В тестовом расчете размер сетки составлял от 50 до 5100 точек, число временных шагов - $1,5 \times 10^5$, каждый временной шаг состоит из порядка 10 итераций. Время выполнения расчетов приведено на рисунке 2.

Из рисунка 2 видно, что на маленьком количестве точек минимальное время выполнения имеет последовательная реализация алгоритма. На втором месте фрагментированная, на третьем – параллельная и на последнем – LuNA. С увеличением количества точек, параллельная реализация сначала догоняет, а потом и обгоняет последовательную реализацию. Это объясняется тем, что с ростом объема вычислений выгода от распараллеливания начинает перевешивать накладные расходы, возникающие вслед-

ствие коммуникаций. При этом время выполнения фрагментированной реализации также больше чем последовательной. Это связано с тем, что вначале каждого итерационного шага алгоритма выделяется память под переменные, а в конце освобождается. Время выполнения LuNA-программы примерно на полтора порядка превышает время других программ. Причиной этого является относительно небольшой размер задачи, и системные накладные расходы на организацию исполнения фрагментированной программы превышают время "полезных"вычислений алгоритма. Тем не менее, несмотря на более низкое качество реализации LuNA-программа проще в разработке, так как система LuNA берет на себя автоматическое обеспечение коммуникаций, распределения ресурсов, управление памятью, и ряд других функций.

Заключение

Применена технология и система фрагментированного программирования LuNA для реализации алгоритма решения одномерной краевой задачи нефть-вода. Выполнено сравнительное тестирование эффективности различных реализаций алгоритма. На основании результатов тестирования можно сделать вывод, что самым оптимальным по времени выполнения является параллельная реализация. Однако реализация в LuNA обладает рядом преимуществ, таких как относительная простота разработки параллельной программы, т.к. программист не должен заботиться о коммуникациях между процессами, распределении ресурсов, управлением памятью, что в общем случае является сложным в реализации.

Литература

[1] *Victor Malyshkin and Vladislav Perepelkin* Optimization methods of parallel execution of numerical programs in the LuNA fragmented programming system // The Journal of Supercomputing, Springer, Volume 61, Number 1 (2012), pp. 235-248. DOI: 10.1007/s11227-011-0649-6.

[2] *Вальковский В.А., Малышкин В.Э.* Синтез параллельных программ и систем на вычислительных моделях. - Новосибирск: Наука. Сиб. отд-ние, 1988. - 129 с.

[3] *М. А. Городничев* Оъединение вычислительных кластеров для крупномасштабного численного моделирования в проекте NumGRID / М. А. Городничев // Весник Новосибирского Государственного Университета (серия Информационные технологии) Том 10, выпуск 4, с.63-73.

[4] *Kireev S.* The LuNA Library of Parallel Numerical Fragmented Subroutines / S. Kireev, V. Malyshkin, H.Fujita // Lecture Notes in Computer Science. - 2011. - Vol. 6873. - P. 290 -301.

[5] *S.Kireev and V.Malyshkin*Fragmentation of numerical algorithms for parallel subroutines library - The J. of Supercomputing, Springer, Springer, Volume 57, Number 2 / August 2011 pp. 161-171.

[6] *Тихонов А. Н., Самарский А. А.* Уравнения математической физики. - 5-е изд.-М.: Наука, 1977. - 736с.

[7] *Самарский А. А., Гулин А. В.* Численные методы: Учеб. пособие для вузов.-М.: Наука. 1989. 432с.

[8] *Бахвалов Н. С, Жидков Н. П., Кобельков Г. М.* Численные методы.- М.: Наука, 1987. - 630с.

[9] *Яненко Н.Н., Коновалов А.Н., Бугров А.Н., Шустов Г.В.* Об организации параллельных вычислений и "распараллеливании"прогонки // Численные методы механики сплошной среды. 9, № 7. Новосибирск, 1978. 139-146.