УДК 004.021, 681.324

С. В. МИНУХИН, канд. техн. наук, проф. ХНЭУ, Харьков;
М. И. СУХОНОС, программист, ООО «Ди Би Бест Технолоджис», Харьков

АЛГОРИТМЫ, ПРОГРАММНАЯ АРХИТЕКТУРА И ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ МОДЕЛИРОВАНИЯ МЕТОДОВ МАСШТАБИРОВАНИЯ СКОРОСТЕЙ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОРОВ ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА

Предложены алгоритмы масштабирования скоростей параллельных процессоров многопроцессорных систем и вычислительных кластеров, имеющих гомогенную и гетерогенную архитектуру. Разработаны программная архитектура и информационная технология для моделирования работы алгоритмов, использующих различные методы распределения заданий на процессоры, приведены примеры и результаты их экспериментального исследования, подтверждающие их эффективность для построения энергоэффективных расписаний выполнения заданий с директивными сроками выполнения в многопроцессорных системах и вычислительных кластерах.

Ключевые слова: процессор, скорость, задание, директивный срок, энергопотребление, длительность, плотность задания, сортировка, загрузка.

Введение. Экономия потребления электроэнергии является важнейшей задачей при разработке и использовании современных процессоров в мобильных устройствах с одноядерными процессорами, серверных фермах и современных ноутбуках с многоядерными процессорами. Оптимизация работы вычислительных кластеров, серверов, многопроцессорных систем и отдельных ПК в настоящее время непосредственно связана с решениями, которые позволяют повысить их экономическую эффективность, в частности, на основе методов оптимизации энергопотребления. На алгоритмическом уровне в настоящее время используется два механизма экономии электроэнергии: масштабирование скорости процессоров и режим включения и отключения процессоров: Процессоры таких производителей как Intel и AMD (технологии Intel SpeedStep и AMD PowerNow) могут работать с переменной скоростью, но при этом, чем выше скорость, тем больше их энергопотребление. Основной задачей масштабирования скорости является такой выбор частот для выполнения заданий, который бы позволил построить энергоэффективный график (расписание) их выполнения, уменьшая при этом время обработки и гарантируя определенную услугу – в допустимом расписании должно отсутствовать запаздывание. Второй механизм экономии электроэнергии заключается в том, что когда процессоры находятся в режиме ожидания, они могут быть в определенный момент переведены в маломощное состоянии «сна». При этом проблемой является определение промежутка

(интервалов) времени, в которые нужно выключать процессоры, принимая во внимание то, что переход обратно в активный режим (включение) требует дополнительных затрат электроэнергии.

Одной из известных в настоящее время технологий для снижения (или оптимизации) потребления электроэнергии первого из приведенных механизмов является динамическое масштабирование скорости (напряжения и частоты, Dynamic Voltage Frequency Scalling) процессора [1–8], при использовании которой процессор может варьировать скорость динамически в зависимости от его загрузки, определяемой параметрами выполняемых заданий. В работах [6, 8] предложены и исследованы алгоритмы масштабирования скорости для одиночных процессоров, показано, что они позволяют строить допустимые расписания для выполнения заданий с различными директивными сроками, в работе [7] предложены алгоритмы масштабирования для гомогенных многопроцессорных систем и вычислительных кластеров на основе предварительной сортировки заданий по их длительности.

Содержательно алгоритмы масштабирования скоростей параллельных процессоров включают следующие этапы:

распределение заданий по процессорам на основе выбранного метода [3, 4, 7];

построение расписания выполнения заданий на каждом процессоре [1, 6–8].

Основной задачей этапа 1 является определение последовательности распределяемых на процессоры заданий таким образом, чтобы оно гарантировало оптимальное по критерию качества расписание их последующего выполнения. В силу полученных в работе [2] результатов в качестве характеристики задания для такой сортировки используется его плотность, которая фактически определяет возможную скорость выполнения задания, если бы оно решалось в автономном режиме. Сортировка плотностей заданий по убыванию (метод Highest Density First, HDF) [3, 4] является наиболее приемлемой, так как отражает механизм выбора скорости процессора, положенного в основу работы [2], позволяющего строить допустимые (отсутствие запаздывания) расписания. Вместе с тем, необходимо отметить, что выбор метода распределения заданий требует дополнительных исследований, позволяющих получить более близкие к оптимальным результаты планирования, связанные с балансировкой загрузки процессоров.

Цель данного исследования – разработать алгоритмы, программное обеспечение и информационные технологии для моделирования методов масштабирования скоростей параллельных гомогенных и гетерогенных процессоров на основе методов с предварительной сортировкой заданий по определенным критериям, их экспериментальное исследование и сравнительный анализ эффективности.

1. Алгоритмы масштабирования скоростей для гомогенных параллельных процессоров

1.1. Алгоритм на основе метода распределения заданий с директивными сроками с сортировкой по убыванию их длительности

Пусть задания имеют директивные сроки выполнения и представляются в виде $t_i = \{L_i, D_i\}$, где L_i – длительность i-го задания, D_i – директивный срок его выполнения (завершения).

Пусть имеется тестовая последовательность заданий с директивными сроками:

$t_1 = \{1, 2\}$, $t_2 = \{1, 3\}$, $t_3 = \{2, 6\}$, $t_4 = \{1, 8\}$, $t_5 = \{3, 9\}$, $t_6 = \{7, 14\}$, $t_7 = \{5, 16\}$, $t_8 = \{4, 22\}$, $t_9 = \{3, 25\}$, $t_0 = \{6, 35\}$.

После сортировки заданий по убыванию их длительности получим:

$t_6 = \{7, 14\}$, $t_{10} = \{6, 35\}$, $t_7 = \{5, 16\}$, $t_8 = \{4, 22\}$, $t_5 = \{3, 9\}$, $t_9 = \{3, 25\}$, $t_3 = \{2, 6\}$, $t_1 = \{1, 2\}$, $t_2 = \{1, 3\}$.

Распределим задания по процессорам в соответствии со следующим алгоритмом.

Шаг 1. Распределяем задания $t_6 = \{7, 14\}$, $t_{10} = \{6, 35\}$, $t_7 = \{5, 16\}$ в порядке убывания их длительности по всем 3 процессорам. Мощность распределяемого множества заданий – 3. В табл. 1–4 в последнем столбце приведены суммарные загрузки каждого из процессоров.

Получаем начальную загрузку процессоров (см. табл. 1):

Таблица 1

Процессор 1	7			7
Процессор 2	6			6
Процессор 3	5			5

Шаг 2. Оставшиеся задания $t_8 = \{4, 22\}$, $t_5 = \{3, 9\}$, $t_9 = \{3, 25\}$, $t_3 = \{2, 6\}$, $t_1 = \{1, 2\}$, $t_2 = \{1, 3\}$ распределяем таким образом, чтобы каждое из нераспределенных заданий, отсортированных по убыванию их длительности, было распределено на наименее загруженный процессор. Мощность распределяемого множества заданий – 3.

Получаем общую загрузку каждого процессора (см. табл. 2):

Таблица 2

Процессор 1	7	3		10
Процессор 2	6	3		9
Процессор 3	5	4		9

Шаг 3. Оставшиеся задания $t_3 = \{2, 6\}$, $t_1 = \{1, 2\}$, $t_2 = \{1, 3\}$ распределяем таким образом, чтобы каждое из нераспределенных заданий, отсортированных по убыванию их длительности, было распределено на

наименее загруженный процессор. Мощность распределяемого множества заданий – 3. Получаем общую загрузку для каждого процессора (см. табл. 3):

Таблица 3

Процессор 1	7	3	1	11
Процессор 2	6	3	2	11
Процессор 3	5	4	1	10

Таким образом, после завершения распределения заданий по процессорам получим следующие расписания выполнения заданий и коэффициенты их средней загрузки после сортировки в порядке возрастания их директивных сроков на каждом из трех процессоров (см. табл. 4):

Таблица 4

Процессор 1	0,5	0,57	0,23	0,23
Процессор 2	0,33	0,37	0,25	0,25
Процессор 3	0,125	0,375	0,45	0,45

Для тестовой последовательности заданий коэффициент средней загрузки [6, 7] составляет 0,94, при этом после распределения заданий между параллельными процессорами должно выполняться неравенство:

$$K_{avr} \leq m^* \max K_{avr\,i} \qquad (1)$$

где max K_{avri} – максимальный коэффициент средней загрузки для всех процессоров распределённых на них заданий;

K_{avr} – коэффициент средней загрузки для всех заданий исходной последовательности – $0,94 \leq 1,35$.

После завершения процедуры распределения заданий по процессорам задания, распределенные на каждый из процессоров, сортируются в порядке возрастания (не убывания) их директивных сроков с последующим построением расписаний выполнения заданий с использованием алгоритмов масштабирования скоростей для одиночных процессоров [6–8].

1.2. Алгоритм на основе метода распределения заданий с директивными сроками с сортировкой по убыванию их плотности

Для описания работы алгоритма и оценки результатов его работы на гомогенных параллельных процессорах используем тестовую последовательность заданий, рассмотренную в п. 1.1:

Без потери общности результатов и для упрощения понимания сути работы алгоритмов выберем количество процессоров равным 3. Отметим, что для корректной работы алгоритмов масштабирования задания отсортированы по возрастанию их директивных сроков.

Рассчитаем плотности заданий d_i для исходной последовательности: d_1=0,5, d_2=0,5, d_3=0,5, d_4=0,25, d_5=0,75, d_6=1,15, d_7=5, d_8=2, d_9=0,5, d_{10}=2.

Отсортируем задания по убыванию (не возрастанию) их плотностей:

d_7=5, d_8=2, d_{10}=2, d_6=1,15, d_5=0,75, d_1=0,5, d_2=0,5, d_3=0,5, d_9=0,5, d_4=0,25.

Отметим, что если директивные сроки заданий совпадают, то в начальном расписании сначала располагается задание с наименьшим директивным сроком выполнения. Далее реализуется следующий алгоритм.

Шаг 1. Распределяем задания в порядке убывания их плотности по всем 3 процессорам. Мощность распределяемого множества заданий – 3. В табл. 5–7 в последнем столбце приведены суммарные загрузки каждого из процессоров.

Получаем начальную загрузку каждого процессора (см. табл. 5):

Таблица 5

Процессор 1	5			5
Процессор 2	2			2
Процессор 3	2			2

Шаг 2. Оставшиеся задания d_6=1,15, d_5=0,75, d_1=0,5, d_2=0,5, d_3=0,5, d_9=0,5, d_4=0,25 распределим таким образом, чтобы каждое из нераспределенных заданий, отсортированных по убыванию их плотности, было назначено на наименее загруженный процессор. Мощность распределяемого множества заданий – 3.

Получаем общую загрузку для каждого процессора (см. табл. 6):

Таблица 6

Процессор 1	5	0,5		5,5
Процессор 2	2	1,15		3,15
Процессор 3	2	0,75		2,75

Шаг 3. Оставшиеся задания d_2=0,5, d_3=0,5, d_9=0,5, d_4=0,25 распределяем таким образом, чтобы каждое из нераспределенных заданий, отсортированных по убыванию их плотности, было распределено на наименее загруженный процессор. Мощность распределяемого множества заданий – 3.

Получаем общую загрузку для каждого процессора (см. табл. 7).

Таблица 7

Процессор 1	5	0,5	0,5	6
Процессор 2	2	1,15	0,5	3,65
Процессор 3	2	0,75	0,5	3,35

Упорядочим полученные расписания по возрастанию директивных сроков выполнения заданий (см. табл. 8).

Таблица 8

Процессор 1	d_1	d_3	d_7	
Процессор 2	d_2	d_6	d_8	
Процессор 3	d_5	d_9	d_{10}	

Рассчитаем коэффициенты средней загрузки процессоров для полученных расписаний выполнения распределенных на них заданий (см. табл. 9).

Таблица 9

Процессор 1	0,5	0,5	0,5	0,5
Процессор 2	0,33	0,57	0,54	0,54
Процессор 3	0,33	0,24	0,34	0,34

Для тестовой последовательности заданий коэффициент средней загрузки составляет 0,94, при этом после распределения заданий между параллельными процессорами выполняется неравенство (1) – $0{,}94 \le 1{,}62$.

После завершения процедуры распределения заданий по процессорам те задания, которые были распределены на каждый из процессоров, сортируются в порядке возрастания (не убывания) их директивных сроков с последующим построением расписаний выполнения заданий с использованием алгоритмов масштабирования скоростей для одиночных процессоров [6–8].

2. Алгоритмы масштабирования скоростей для гетерогенных параллельных процессоров

Для гетерогенной многопроцессорной системы задача масштабирования скорости процессоров может быть решена следующим образом: для известных скоростей m процессоров необходимо:

выполнить нормализацию скоростей каждого из m процессоров по отношению к максимальной скорости процессора из их множества;

учитывая, что время выполнения задания t_i на процессоре обратно пропорционально его скорости s_i, т.е.

$$\frac{t_k}{t_l} = \frac{s_l}{s_k},$$

где $s_l > s_k$, $t_k > t_l$, получаем, что время выполнения задания t^* на любом процессоре, имеющем скорость s_k, по отношению ко времени выполнения задания на процессоре с максимальной скоростью s_{max} определяется по формуле:

$$t_i^* = \frac{t_i}{\frac{s_k}{s_{\max}}} = t_i \frac{s_{\max}}{s_k}.$$

2.1. Алгоритм на основе метода распределения заданий с директивными сроками с сортировкой по убыванию их плотности (HDF)

Для описания работы алгоритма и оценки результатов его работы на гетерогенных параллельных процессорах используем следующую тестовую последовательность: $t_1 = \{1, 2\}$, $t_2 = \{1, 3\}$, $t_3 = \{2, 6\}$, $t_4 = \{1, 8\}$, $t_5 = \{3, 9\}$, $t_6 = \{7, 14\}$, $t_7 = \{5, 16\}$, $t_8 = \{4, 22\}$, $t_9 = \{3, 25\}$, $t_{10} = \{6, 35\}$.

Без потери общности полученных результатов и для упрощения понимания работы алгоритмов выберем количество процессоров равным 3. Отметим, что для корректной работы алгоритмов масштабирования задания отсортированы по возрастанию их директивных сроков. Рассчитаем плотности заданий для исходной последовательности и отсортируем задания по убыванию (не возрастанию) их плотностей (см. п.1.2): d_7=5, d_8=2, d_{10}=2, d_6=1,15, d_5=0,75, d_1=0,5, d_2=0,5, d_3=0,5, d_9=0,5, d_4=0,25. Отметим, что если директивные сроки заданий совпадают, то в начальном расписании сначала располагается задание с наименьшим директивным сроком выполнения.

Шаг 1. Распределяем задания в порядке убывания их плотности по всем 3-ем процессорам с учетом их разных скоростей. Мощность распределяемого множества заданий – 3. В табл. 10–12 в последнем столбце приведены суммарные значения плотностей заданий, распределенных на каждый процессор. Получаем начальную загрузку каждого процессора (см. табл. 10).

Таблица 10

Процессор 1	5			5
Процессор 2	2,4			2,4
Процессор 3	4			4

Шаг 2. Оставшиеся задания d_6=1,15, d_5=0,75, d_1=0,5, d_2=0,5, d_3=0,5, d_9=0,5, d_4=0,25 распределяем таким образом, чтобы каждое из нераспределенных заданий, отсортированных по убыванию их плотности, было распределено на наименее загруженный процессор. Мощность распределяемого множества заданий – 3. Получаем общую загрузку каждого процессора (см. табл. 11).

Таблица 11

Процессор 1	5	0,5		5,5
Процессор 2	2,4	1,4		3,8
Процессор 3	4	1		5

Шаг 3. Оставшиеся задания d_2=0,5, d_3=0,5, d_9=0,5, d_4=0,25 распределяем таким образом, чтобы каждое из нераспределенных заданий, отсортированных по убыванию их плотности, было распределено на наименее загруженный процессор. Мощность распределяемого множества заданий – 3. Получаем общую загрузку каждого процессора (см. табл. 12).

Таблица 12

Процессор 1	5	0,5	0,5	6
Процессор 2	2,4	1,4	0,6	4,4
Процессор 3	4	1	1	6

Упорядочим полученные для процессоров расписания по возрастанию директивных сроков выполнения заданий (см. табл. 13).

Таблица 13

Процессор 1	d_1	d_3	d_7	
Процессор 2	d_2	d_6	d_8	
Процессор 3	d_5	d_9	d_{10}	

Рассчитаем коэффициенты средней загрузки процессоров для полученных для них расписаний выполнения заданий (см. табл. 14).

Таблица 14

Процессор 1	0,5	0,5	0,7	0,57
Процессор 2	0,33	0,57	0,54	0,48
Процессор 3	0,33	0,24	0,34	0,31

Для тестовой последовательности заданий коэффициент средней загрузки составляет 0,94, при этом после распределения заданий между параллельными процессорами выполняется неравенство (1) – 0,94 ≤ 1,71. После завершения процедуры распределения всех заданий для каждого процессора реализуются алгоритмы масштабирования скорости (напряжения и частоты) одиночного процессора, рассмотренные в работах [6–8].

Общая временная сложность данного алгоритма распределения заданий составляет: сложность определения плотностей заданий и сортировки заданий по убыванию плотностей – **O**(n + $n$log$n$); сложность определения процессора с максимальной скоростью и сортировки скоростей процессоров по их убыванию – **O**($m$log$m$); сложность определения процессора с наименьшей загрузкой с сортировкой процессоров по возрастанию их загрузки – **O**($m$log$m$); сложность масштабирования скорости и определения общей загрузки на каждом этапе для каждого из m процессоров – **O**($n$log$n$ + $m$log$m$ + m*k + $m$log$m$), где k – количество заданий, распределенных на один процессор. Таким образом, временная сложность данного алгоритма составляет **O**($n$log$n$ + $m$log$m$ + m*k), т.е. является полиномиальной.

2.2. Алгоритм на основе метода распределения заданий с директивными сроками с сортировкой заданий по убыванию их плотности с использованием процедуры Round Robin (HDF RR)

Предлагаемый алгоритм является модификацией алгоритмов, исследованных в работах [3, 4], и отличается тем, что в нем в нем используются нормализованные (приведенные) относительно максимального значения скорости процессоров.

Шаг 1. Распределяем тестовые задания в порядке убывания их плотности (см. п. 1.2) по всем 3-ем процессорам с учетом их разных скоростей. Мощность распределяемого множества заданий – 3. В табл. 15–17 в последнем столбце приведены суммарные значения плотностей заданий загрузок процессоров. Получаем начальную загрузку каждого процессора (см. табл. 15).

Таблица 15

Процессор 1	5			5
Процессор 2	2,4			2,4
Процессор 3	4			4

Шаг 2. Оставшиеся задания d_6=1,15, d_5=0,75, d_1=0,5, d_2=0,5, d_3=0,5, d_9=0,5, d_4=0,25 распределяем таким образом, чтобы каждое из нераспределенных заданий, отсортированных по убыванию их плотности, было распределено на следующий в списке процессор. Мощность распределяемого множества заданий – 3.

Получаем общую загрузку каждого процессора (см. табл. 16).

Таблица 16

Процессор 1	5	1,15		6,15
Процессор 2	2,4	0,9		3,3
Процессор 3	4	1		5

Шаг 3. Оставшиеся задания d_2=0,5, d_3=0,5, d_9=0,5, d_4=0,25 распределяем таким образом, чтобы каждое из нераспределенных заданий, отсортированных по убыванию их плотности, было распределено на следующий в списке процессор. Мощность распределяемого множества заданий – 3. Получаем общую загрузку каждого процессора (см. табл. 17).

Таблица 17

Процессор 1	5	1,15	0,5	6,65
Процессор 2	2,4	0,9	0,6	3,9
Процессор 3	4	1	1	6

Упорядочим полученные расписания для каждого из процессоров по возрастанию директивных сроков выполнения распределенных на них заданий (см. табл. 18).

Таблица 18

Процессор 1	d_2	d_6	d_7	
Процессор 2	d_3	d_5	d_8	
Процессор 3	d_1	d_9	d_{10}	

Рассчитаем коэффициенты средней загрузки процессоров для полученных расписаний выполнения заданий (см. табл. 19).

Таблица 19

Процессор 1	0,33	0,57	0,81	0,57
Процессор 2	0,33	0,45	0,36	0,38
Процессор 3	0,5	0,16	0,28	0,31

Как следует из полученных результатов, коэффициенты средней загрузки процессоров составляют 0,57, 0,38 и 0,31, что соответствует худшей балансировке загрузки процессоров по сравнению с предыдущим методом. После завершения процедуры распределения всех заданий для каждого процессора реализуются алгоритмы масштабирования скорости одиночного процессора, рассмотренные в работах [6–8].

3. Программная архитектура и информационная технология реализации и экспериментального исследования алгоритмов

Для экспериментального исследования предложенных алгоритмов разработана программа на языке C# на платформе .NET Framework 4.0, состоящая из шести модулей (см. рис. 1). Модуль EDFScheduler.Model содержит описание всех сущностей системы. Модуль EDFScheduler.Data отвечает за работу с хранилищем данных (БД). Модуль EDFScheduler.Core содержит реализацию всех алгоритмов планирования, распределения, аппроксимации, сбор статистики для визуализации полученных результатов. Модуль EDFScheduler.Graphics предоставляет компоненты для построения различных графических данных. Модуль EDFScheduler.Resources определяет локализацию приложения для различных языков. Модуль EDFScheduler.UI содержит компоненты пользовательского интерфейса, а также приводит в действие остальные модули.

Диаграмма классов программы приведена на рис. 2. На рис. 2 показаны следующие модули: Main Window – окно с пользовательским интерфейсом; IProcessorDistributor – интерфейс, распределяющий процессорные элементы по последовательностям и пакетам работ (моделирование гетерогенной системы); IGenerator – интерфейс генерации последовательности работ и их пакеты; IJobDistributor – интерфейс распределения работ по различным процессорным элементам; IPowerMeter – интерфейс подсчета количества энергии; IProcessorRepository – интерфейс работы с хранилищем данных (БД) процессоров; IDVSAlgorithm – интерфейс алгоритмов планирования переключения режимов работы процессора; IApproximateAlgorithm –

интерфейс алгоритмов аппроксимации относительных скоростей процессоров к возможным (рассчитываемым) дискретным режимам их работы.

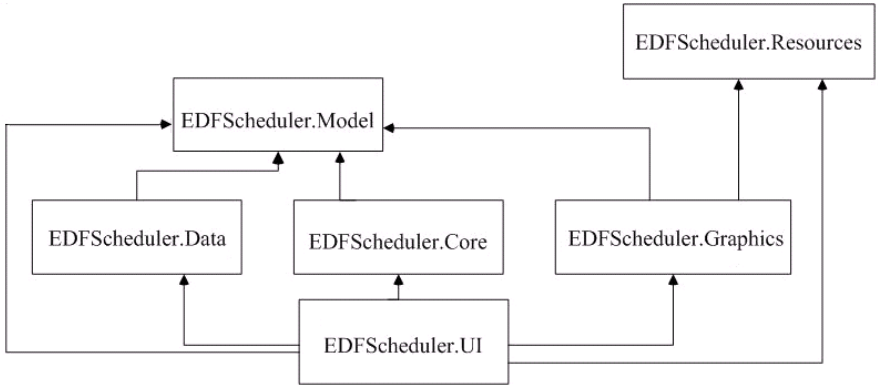

Рис. 1 – Структура программного обеспечения реализации и моделирования работы алгоритмов

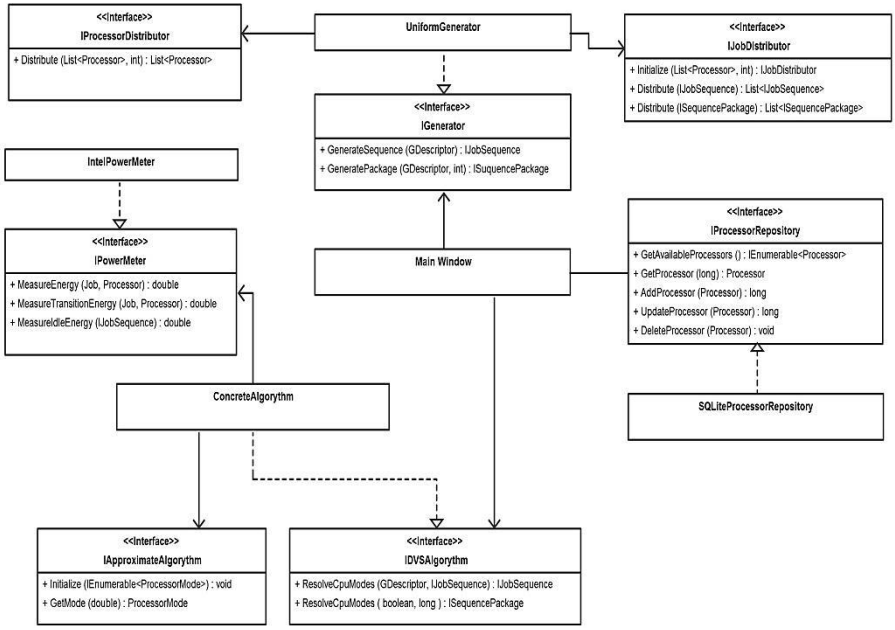

Рис. 2 – Диаграмма классов программы

Структура БД программы приведена на рис. 3, на котором приняты следующие обозначения: таблица «Processor» хранит в себе информацию о

процессоре, а именно: Id – идентификатор, Name – имя, Capacitance – емкостное сопротивление; TransitionLatency – максимальное время, которое может быть затрачено на переключение; IdleCoefficient – отношение количества электроэнергии, затрачиваемого в холостом режиме относительно электроэнергии, потребляемой в режиме с наибольшей скоростью; NominalModelId – идентификатор номинального режима; таблица «ProcessorMode» содержит информацию о режимах процессора: Id – идентификатор, Processor – идентификатор родительского процессора, Name – имя, Frequency – частота работы процессора в данном режиме, Voltage – напряжение ядра процессора в данном режиме, RelativeSpeed – относительная скорость процессора; таблица «Setting» содержит общие настройки системы.

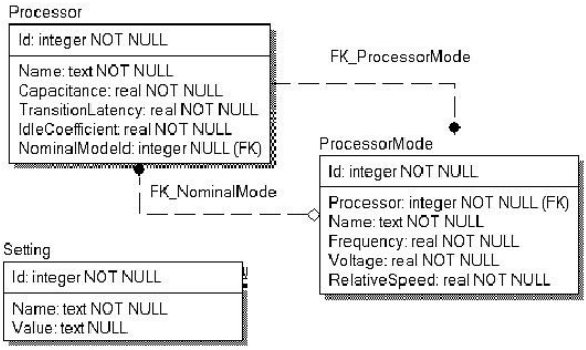

Рис. 3 – Структура БД программной реализации алгоритмов масштабирования скоростей процессоров

4. Результаты вычислительных экспериментов и их анализ

Экспериментальное исследование работы алгоритмов HDF и HDF RR проводилось на основе следующих настроек: длительности заданий определялись по равномерному закону в диапазоне [1, 1000]; директивные сроки выполнения заданий – по равномерному закону в диапазоне [1, 3]; количество заданий – в диапазонах [1, 100], [1, 1000], [1, 10000]; количество процессоров – в диапазоне 1–100, количество наблюдений – 50. В качестве критерия (параметра) балансировки загрузки процессоров для методов распределения использовано стандартное (среднеквадратическое) отклонение загрузок всех процессоров от среднего значения. Результаты моделирования алгоритмов масштабирования для различных методов показаны на рис. 4–7.

Выводы. Рассмотренные в работе алгоритмы масштабирования частот для параллельных процессоров и их моделирование при помощи разработанного программного обеспечения позволяют реализовать методы распределения заданий между процессорами и последующую настройку режимов

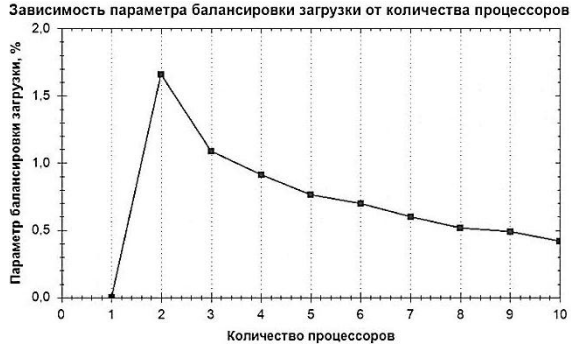

Рис. 4 – Зависимость балансировки загрузки от количества процессоров (типов процессоров – 2) для метода HDF (1000 заданий)

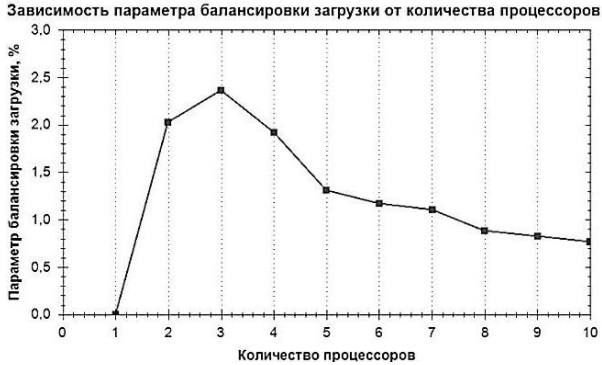

Рис. 5 – Зависимость балансировки загрузки от количества процессоров (типов процессоров – 3) для метода HDF (2000 заданий)

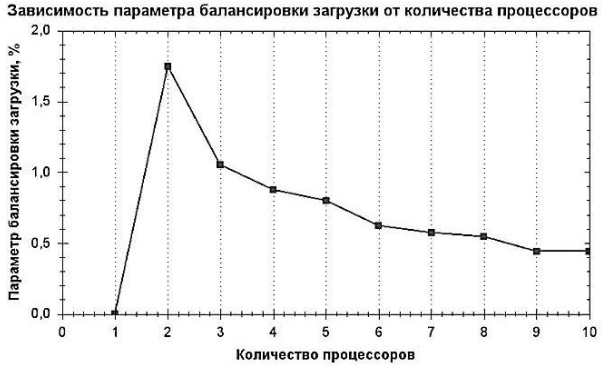

Рис. 6 – Зависимость балансировки загрузки от количества процессоров (типов процессоров – 2) для метода HDF RR (1000 заданий)

Рис. 7 – Зависимость балансировки загрузки от количества процессоров (типов процессоров – 3) для метода HDF RR (2000 заданий)

переключения скоростей одиночных процессоров, обеспечивающих построение допустимых расписаний выполнения заданий и, таким образом, оптимизацию их энергопотребления.

Полученные экспериментальные результаты моделирования вычислительных систем с различным количеством типов (моделей) процессоров, характеризующих гетерогенность архитектуры многопроцессорный системы, их количеством, различным количеством заданий и директивных сроков их выполнения показали, что предлагаемые методы позволяют достичь лучшей балансировки загрузки по сравнению с существующими в случае, когда количество заданий существенно превышает количество процессоров; гетерогенность многопроцессорной системы ухудшает балансировку загрузки; неравномерная загрузка процессоров может привести к ухудшению результатов работы алгоритмов масштабирования скоростей одиночных процессоров. В дальнейшем предполагается разработка алгоритмов распределения заданий между параллельными процессорами, в которых балансировка загрузки осуществляется на каждом шаге распределения заданий на основе определения текущей величины средней загрузки процессора, то есть после каждого распределения заданий на процессоры.

Список литературы: 1. *Kim K. H.* Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters / *K. H. Kim, R. Buyya, J. Kim* // Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid. – 2007. – p. 541–548. **2.** *Yao F.F.* A scheduling model for reduced CPU energy / *F.F. Yao, A.J. Demers, S. Shenker* // Proc. 36th IEEE Symposium on Foundations of Computer Science, 1995. – p. 374–382. **3.** *Albers S.* Speed scaling on parallel processors / *S. Albers, F. Müller, S. Schmelzer* // Proc. 19th ACM Symposium on Parallelism in Algorithms and Architectures. – 2007. – p. 289–298. **4.** *Angel E.* Speed scaling on parallel processors with migration / *E. Angel, E. Bampis, F. Kacem [et. al.]* // Proc. Of International Conference Euro-Par 2012 Parallel Processing. – 2012. – p. 128–140. **5.** *Lefurgy C.* Energy Management for Commercial Servers / *C. Lefurgy, K. Rajamani, F. Rawson et al.* // Computer. – 2003. – №36 (12). – p. 39–48. **6.** *Минухин С.В.* Алгоритмы оптимизации энергопотребления и повышения эффективности процессоров с масштабированием частоты и напряжения гетерогенного кластера / *С.В. Минухин,*

М.И. Сухонос // International Conference Parallel and Distributed Computing Systems PDCS 2013 (Ukraine, Kharkiv, March 13-14, 2013). – 2013. – p. 209–217. **7.** *Минухин С. В*. Энергоэффективные алгоритмы масштабирования скорости процессоров вычислительного кластера / *С. В. Минухин* // Second International Conference "Cluster Computing" CC 2013 (Ukraine, Lviv, June 3-5, 2013). – 2013. – С. 131–140. **8.** *Минухин С. В.* Алгоритмы оптимального определения напряжения и частоты процессоров вычислительного кластера для пакетов заданий с директивными сроками выполнения / *С. В. Минухин* // Проблеми розвитку ІТ-індустрії. Системи обробки інформації. – 2013. – Вип. №3 (110). – Т. 2. – С. 184.

Надійшла до редколегії 28.08.2013

УДК 004.021, 681.324

Алгоритмы, программная архитектура и информационная технология моделирования методов масштабирования скоростей параллельных процессоров вычислительного кластера // С. В. Минухин, М. И. Сухонос // Вісник НТУ «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – Х.: НТУ «ХПІ». – 2013. – № 62 (1035)**. –** С. 78–92. – Бібліогр.: 8 назв.

Запропоновано алгоритми масштабування швидкостей паралельних процесорів багатопроцесорних систем і обчислювальних кластерів, що мають гомогенну та гетерогенну архітектуру. Розроблено програмну архітектура та інформаційну технологію для моделювання роботи алгоритмів, що використовують різні методи розподілу завдань на процесори, наведено приклади та результати їх експериментального дослідження, які підтвердили їх ефективність для побудови енергоефективних розкладів виконання завдань з директивними строками виконання в багатопроцесорних системах і обчислювальних кластерах.

Ключові слова: алгоритм, процесор, швидкість, завдання, директивний строк, енергоспоживання, тривалість, щільність завдання, сортування, завантаження.

There have been proposed the algorithms for speed scaling on parallel processors for multiprocessor systems and computing clusters with homogeneous and heterogeneous architectures. Software architecture and information technology for simulation of algorithms using a variety of methods for allocating tasks to processors are developed. There are examples and the results of their experimental studies proving its effectiveness for constructing energy efficient scheduling task execution with deadlines in multiprocessor systems and computing clusters.

Keywords: algorithm, processor, speed, task, deadline, energy consumption, duration, density task, sorting, load.