Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ РОБОТОТЕХНИКА И КОМПЛЕКСНАЯ АВТОМАТИЗАЦИЯ (РК)

КАФЕДРА РК6 «СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ»

Отчет по лабораторной работе

Исследование эффективности динамической балансировки загрузки МВС с использованием имитационного моделирования

Студент ________________ **Абидоков Р. Ш.**
РК6-21М

Преподаватель ________________ **Карпенко А. П.**

2022 г.

Постановка задачи исследования эффективности балансировки загрузки МВС

Пусть $X - n$-мерный вектор параметров задачи. Положим, что $X \in R^n$, где $R^n - n$-мерное арифметическое пространство. Параллелепипедом допустимых значений вектора параметров назовем не пустой параллелепипед $\Pi = \{X \mid x_i^- \leq x_i \leq x_i^+, i \in [1,n]\}$, где x_i^-, x_i^+ – заданные константы. На вектор X дополнительно наложено некоторое количество функциональных ограничений, формирующих множество $D = \{X \mid g_i(X) \geq 0, j = 1,2, \dots\}$, где $g_i(X)$ – непрерывные ограничивающие функции.

На множестве $D_x = \Pi \cap D$ тем или иным способом (аналитически или алгоритмически) определена вектор-функция $\mathrm{F}(X)$ со значениями в пространстве R^m. Ставится задача поиска значения некоторого функционала $\Phi(\mathrm{F}(X))$.

Положим, что приближенное решение поставленной задачи может быть найдено по следующей схеме:

Шаг 1. Покрываем параллелепипед П некоторой сеткой Ω (равномерной или неравномерной, детерминированной или случайной) с узлами $X_1, X_2 \dots X_z$.

Шаг 2. В тех узлах сетки Ω, которые принадлежат множеству D_x, вычисляем значения вектор функции $\mathrm{F}(X)$.

Шаг 3. На основе вычисленных значений вектор функции $\mathrm{F}(X)$ находим приближенное значение функционала $\Phi(\mathrm{F}(X))$.

Суммарное количество арифметических операций, необходимых для *однократного* определения принадлежности вектора X множеству D_x (т.е. суммарную вычислительную сложность ограничений $x_i^- \leq x_i \leq x_i^+$ и ограничивающих функций $g_i(X)$, обозначим $C_g \geq 0$. *Далее в эксперименте будем полагать* $C_g = 0$.

Неизвестную вычислительную сложность вектор-функции $\mathrm{F}(X)$ обозначим $\mathrm{C}_f(X)$. Подчеркнем зависимость величины C_f от вектора X. Величина $\mathrm{C}_f(X)$ удовлетворяет, во-первых, очевидному ограничению $\mathrm{C}_f(X) \geq 0$. Во-вторых, положим, что известно ограничение сверху на эту величину C_f^{max}, имеющее смысл ограничения на максимально допустимое время вычисления значения $\mathrm{F}(X)$. Вычислительную сложность $\mathrm{C}_f(X_i)$ назовем вычислительной сложностью узла $X_i, i \in [1, Z]$.

Вычислительную сложность генерации сетки Ω положим равной ZC_Ω, а вычислительную сложность конечномерной аппроксимации функционала $\Phi(\mathrm{F}(X))$ - равной ζC_Ω, где ζ (дзета) - общее количество узлов сетки Ω, принадлежащих множествуD_x.

Далее в эксперименте также будем полагать $C_\Omega = 0, C_\Phi = 0$.

В качестве вычислительной системы рассмотрим однородную МВС с распределенной памятью, состоящую из процессоров $P_1, P_2 \dots P_N$ и *host*-процессора, имеющих следующие параметры:

- t – время выполнения одной арифметической операции с плавающей запятой;
- $d = d(N) -$ диаметр коммуникационной сети;
- $l -$ длина вещественного числа в байтах;
- t_s – латентность коммуникационной сети;
- t_c – время передачи байта данных между двумя соседними процессорами системы без учета времени t_s.

В качестве меры эффективности параллельных вычислений используем ускорение

$$S_i(N) = \frac{T(1)}{T_i(N)}, \qquad (1)$$

где $T(1) -$ время последовательного решения задачи на одном процессоре системы, $T_i(N) -$ время параллельного решения той же задачи на N процессорах, $i = 1,2 -$ номер метода балансировки.

Динамическая балансировка загрузки

Положим, что из числа Z узлов расчетной сетки Ω множеству D_x принадлежит ζ узлов $\tilde{X}_1, \tilde{X}_2 \dots \tilde{X}_\zeta$. Разобьем эти узлы на $K, K \geq N$ непересекающихся подмножеств $\overline{\omega_\iota}, i \in [1:K]$ и для простоты примем, что величины ζ, K кратны, так что каждое подмножество $\overline{\omega_\iota}$ содержит $z = \frac{\zeta}{K}$ узлов. Совокупность подмножеств $\overline{\omega_\iota}$ обозначим $\{\overline{\omega}\}$. Правило построения совокупности $\{\overline{\omega}\}$ приведено на Рис. 1.

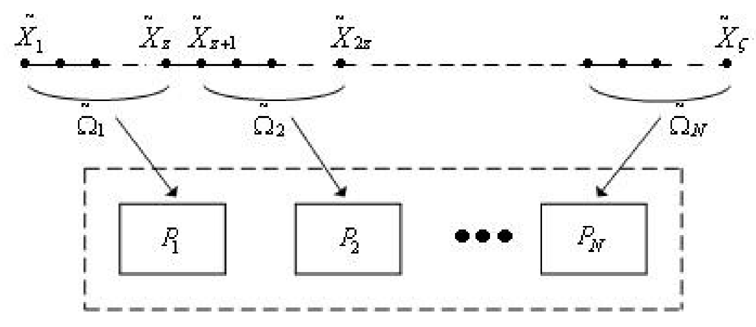

Рис. 1. Схема построения совокупности подмножеств $\{\overline{\omega}\}$

Схема параллельного решения поставленной задачи с использованием динамической равномерной балансировки загрузки имеет следующий вид:

Шаг 1. Host-процессор выполняет следующие действия:

- строит сетку Ω;
- среди всех узлов $X_1, X_2 \dots X_Z$ сетки Ω выделяет узлы $\tilde{X}_1, \tilde{X}_2 \dots \tilde{X}_\zeta$;
- разбиваем узлы $\tilde{X}_1, \tilde{X}_2 \dots \tilde{X}_\zeta$ на K подмножеств $\overline{\omega_\iota}$, $i \in [1:K]$;
- посылает процессору P_i, $i \in [1:N]$ координаты узлов первого из нераспределенных подмножеств $\overline{\omega_\iota}$.

Шаг 2. Процессор P_i выполняет следующие действия:

- принимает от host-процессора координаты узлов подмножества $\overline{\omega_\iota}$;
- вычисляет во всех узлах подмножества $\overline{\omega_\iota}$ значения вектор-функции $F(x)$;
- посылает host-процессору вычисленные значения $F(x)$;

Шаг 3. Если исчерпана не вся совокупность подмножеств $\{\overline{\omega}\}$, то host-процессор посылает, а процессор P_i принимает координаты следующего подмножества из указанной совокупности узлов, которое обрабатывается процессором P_i аналогично шагу 2 и т.д.

Шаг 5. Если исчерпаны все подмножества совокупности $\{\overline{\omega}\}$, то host-процессор:

- посылает освободившемуся процессору P_i сообщение об окончании решения задачи;
- после получения всех вычисленных значений функции $F(x)$ от всех процессоров вычисляет приближенное значение функционала $\Phi(F(x))$;

Экспериментальная часть

Рассмотрим двумерную задачу ($n = 2$). Параллелепипед Π в этом случае представляет собой прямоугольник $\Pi = \{X | x_i^- \le x_i \le x_i^+, i \in [1,2]\}$. Положим, что $x_1^- = x_2^- = 0, x_1^+ = x_2^+ = 1$, так что область Π является квадратом (Рис. 2).

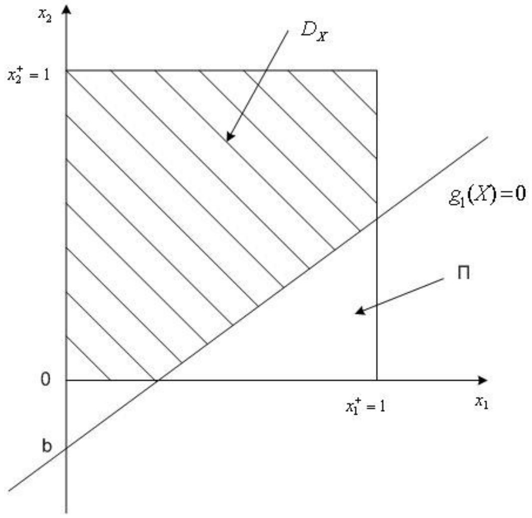

Рис. 2. Расчетная область задачи

Множество D формируется с использованием одной ограничивающей функции $g_1(X) \geq 0$, то есть $D = \{X | g_1(X) \geq 0\}$. Примем, что эта функция линейна и проходит через заданную преподавателем точку плоскости $O(x_1, x_2)$ с координатами $(0, b)$, как показано на Рис. 2.

Таким образом, уравнение этой функции имеет вид $x_2 = ax_1 + b,\ a > 0$ В соответствии с номером варианта заданы значения параметров ограничивающей функции: $a = 1.0,\ b = -0.1$. Общее количество узлов $Z = 256 * 256 = 65536$, количество попавших в область D_x узлов $\zeta = 38971$ (принимаем равным 39168, как ближайшим кратным 256).

Параметры моделируемой МВС:

$N = 2,4,8,16,32,64,128,256$;

$C_f = 10^2, 10^3, 10^4$ с;

$m = 100$;

$l = 8$;

$t = 10 * 10^{-9}$ с;

$t_s = 50 * 10^{-6}$ с;

$t_c = \frac{1}{80} * 10^{-6}$ с;

$d(N) = 2\sqrt{N} - 1$;

Полученные значения матожиданий и среднеквадратичных отклонений значений ускорений для целевой функции со сложностями, равномерно распределенными в интервалах $C_f \in [0, C_f^{max}]$, $C_f^{max} = 2 * 10^4, 2 * 10^6$, приведены в Табл. 1, 2. Исходный код программы приведен в Приложении 1.

Табл. 1. Полученные результаты для различных N при $r = 1$

N	$S,\ C_f^{max} = 2 \cdot 10^4$	$S, C_f^{max} = 2 \cdot 10^6$
2	1.54	1.59
4	2.90	3.17
8	5.34	6.36
16	9.69	12.65
32	16.98	24.99
64	28.82	47.76
128	47.99	95.85
256	74.99	182.07

Табл. 2. Полученные результаты для различных N при $r = 300$

N	$S,\ C_f^{max} = 2 \cdot 10^4$		$S, C_f^{max} = 2 \cdot 10^6$	
	$M^*[S(N)]$	$\sigma^*[S(N)]$	$M^*[S(N)]$	$\sigma^*[S(N)]$
2	1.532	0.006	1.582	0.006
4	2.890	0.013	3.179	0.014
8	5.341	0.025	6.324	0.032
16	9.661	0.042	12.538	0.070
32	16.931	0.095	24.726	0.173
64	28.882	0.178	48.297	0.428
128	47.593	0.291	93.869	1.002
256	75.202	0.439	180.611	2.307

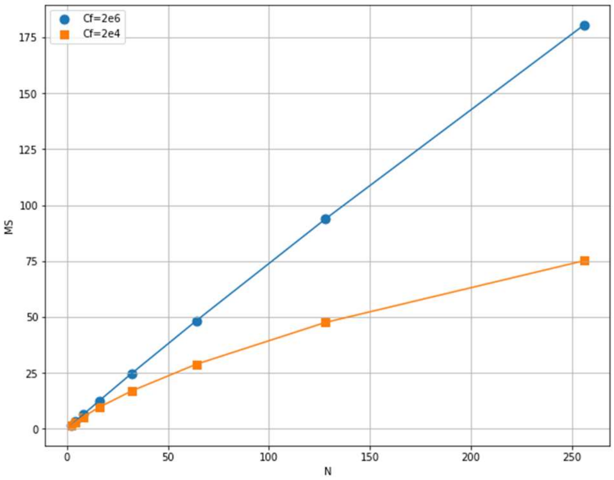

Рис. 3. График $M^*[S(N)]$

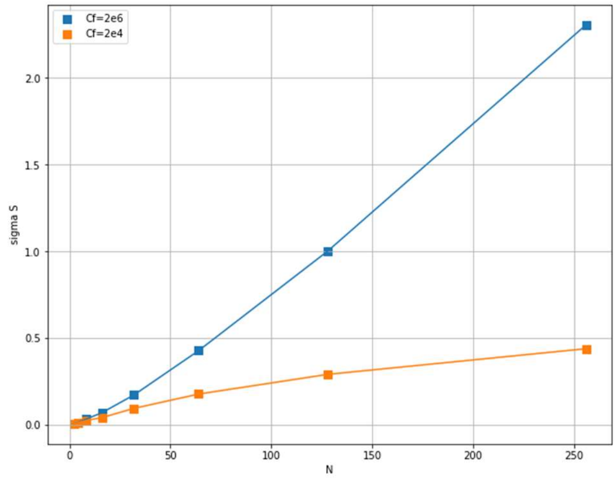

Рис. 4. График $\sigma^*[S(N)]$

Ответы на контрольные вопросы

1. *Чем объясняется наблюдаемый характер зависимостей ускорений* $S(N)$*, оценки математического ожидания ускорения* $M^*[S(N)]$ *и оценки среднего квадратического отклонения* $\sigma^*[S(N)]$*?*

С ростом числа процессоров уменьшается количество полезных вычислений, совершаемых каждым процессором, при этом издержки на коммуникацию не уменьшаются – как следствие, уменьшается эффективность распараллеливания.

2. *Чем объясняется наблюдаемый характер зависимостей ускорений* $S(N)$*, оценки математического ожидания ускорения* $M^*[S(N)]$ *и оценки среднего квадратического отклонения* $\sigma^*[S(N)]$ *от величины* C_f^{max}?

С увеличением величины C_f^{max} уменьшается доля времени, затрачиваемого на коммуникацию между процессами, и увеличивается доля времени, затрачиваемого на вычисление функции в узлах, которое и делится между процессорами – как следствие, растет эффективность распараллеливания.

Исходный код программы

```
N_proc EQU 256
points_N EQU 39168
t_s EQU 50e-6
m_s EQU 100
l_s EQU 8
t_c EQU 0.125e-7
N_gr EQU 2
d_s EQU SQR(N_proc)-1
k_1 EQU 4
s_1 EQU (points_N/N_proc)/k_1
T_ik EQU 2

t EQU 1e-8
uniform_cf_par FUNCTION rn2,c2
0,0.0/1,2e4
uniform_cf_posl FUNCTION rn3,c2
0,0.0/1,2e4

proc_par STORAGE 256
proc_posl STORAGE 1

us VARIABLE p4/p3

tabl_s TABLE v$us,42,0.25,60

generate 1e8,100
split (N_proc - 1)

queue qhost1_par
seize host
depart qhost1_par
advance 5e-6,3e-6
release host

assign 1,s_1
assign 5,k_1

queue qproc_par
enter proc_par
depart qproc_par;

proc3 advance T_ik,1e-8
proc2 advance t,fn$uniform_cf_par
loop 1,proc2
assign 1,s_1
loop 5,proc3
leave proc_par

queue qhost2_par
seize host
depart qhost2_par
advance 5e-6,3e-6
release host
assemble (N_proc)
assign 3,m1

* последовательная обработка
mark 2
split (points_N - 1)

queue qhost1_posl
seize host
depart qhost1_posl
advance 5e-6,3e-6
release host

queue qproc_posl
enter proc_posl
depart qproc_posl
advance t,fn$uniform_cf_posl
leave proc_posl

queue qhost2
seize host
depart qhost2
advance 5e-6,3e-6
release host

assemble points_N
assign 4,mp2

tabulate tabl_s

TERMINATE 1

START 300
```