|  |  |
| --- | --- |
|  | **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 статистической балансировки загрузки МВС**

Пусть -мерный вектор параметров задачи. Положим, что  
 , где -мерное арифметическое пространство. Параллелепипедом допустимых значений вектора параметров назовем не пустой параллелепипед  
где заданные константы. На вектор дополнительно наложено некоторое количество функциональных ограничений, формирующих множество непрерывные ограничивающие функции.

На множестве тем или иным способом (аналитически или алгоритмически) определена вектор-функция со значениями в пространстве . Ставится задача поиска значения некоторого функционала .

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

*Шаг 1.*Покрываем параллелепипед некоторой сеткой Ω (равномерной или неравномерной, детерминированной или случайной) с узлами *Шаг 2.* В тех узлах сетки Ω, которые принадлежат множеству **,** вычисляем значения вектор функции .

*Шаг 3.* На основе вычисленных значений вектор функции находим приближенное значение функционала.

Суммарное количество арифметических операций, необходимых для *однократного* определения принадлежности вектора множеству (т.е. суммарную вычислительную сложность ограничений и ограничивающих функций , обозначим . *Далее в эксперименте будем полагать .*

Неизвестную вычислительную сложность вектор-функции обозначим . Подчеркнем зависимость величины  от вектора *X*. Величина удовлетворяет, во-первых, очевидному ограничению . Во-вторых, положим, что известно ограничение сверху на эту величину , имеющее смысл ограничения на максимально допустимое время вычисления значения . Вычислительную сложность назовем вычислительной сложностью узла.

Вычислительную сложность генерации сетки Ω положим равной , а вычислительную сложность конечномерной аппроксимации функционала - равной, где (дзета) - общее количество узлов сетки Ω, принадлежащих множеству**.**

*Далее в эксперименте также будем полагать .*

В качестве вычислительной системы рассмотрим однородную МВС с распределенной памятью, состоящую из процессоров и *host*-процессора, имеющих следующие параметры:

* – время выполнения одной арифметической операции с плавающей запятой;
* диаметр коммуникационной сети;
* длина вещественного числа в байтах;
* – латентность коммуникационной сети;

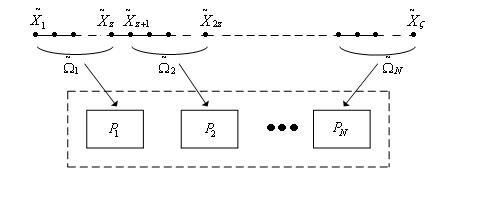
* – время передачи байта данных между двумя соседними процессорами системы без учета времени .

В качестве меры эффективности параллельных вычислений используем ускорение

где время последовательного решения задачи на одном процессоре cистемы, время параллельного решения той же задачи на *N* процессорах, номер метода балансировки.

**Статическая балансировка загрузки методом равномерной декомпозиции параллелепипеда П**

Положим, что из числа узлов расчетной сетки Ω множеству принадлежит узлов. Разобьем эти узлы на непересекающихся подмножеств и для простоты примем, что величины кратны, так что каждое подмножество содержит узлов. Совокупность подмножеств обозначим . Правило построения совокупности приведено на Рис. 1.



**Рис. 1**. Схема построения совокупности подмножеств

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 сетки Ω выделяет узлы ;
* разбиваем узлы на подмножеств ;
* посылает процессору координаты узлов первого из нераспределенных подмножеств .

*Шаг 2*. Процессор выполняет следующие действия:

* принимает от host-процессора координаты узлов подмножества ;
* вычисляет во всех узлах подмножества значения вектор-функции ;
* посылает host-процессору вычисленные значения ;

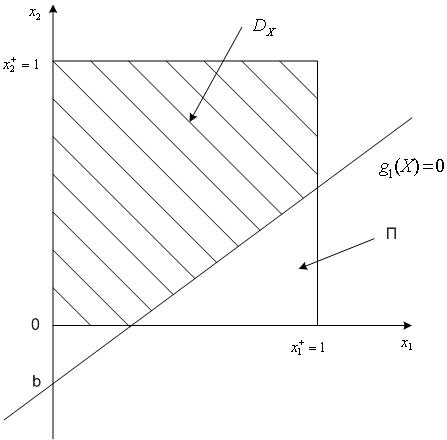
*Шаг 3*. Если исчерпана не вся совокупность подмножеств , то host-процессор посылает, а процессор принимает координаты следующего подмножества из указанной совокупности узлов, которое обрабатывается процессором аналогично шагу 2 и т.д.

*Шаг 5*. Если исчерпаны все подмножества совокупности , то host-процессор:

* посылает освободившемуся процессору сообщение об окончании решения задачи;
* после получения всех вычисленных значений функции от всех процессоров вычисляет приближенное значение функционала ;

**Экспериментальная часть**

Рассмотрим двумерную задачу (). Параллелепипед П в этом случае представляет собой прямоугольник . Положим, что , так что область П является квадратом (Рис. 2).



**Рис. 2**. Расчетная область задачи

Множество *D* формируется с использованием одной ограничивающей функции , то есть . Примем, что эта функция линейна и проходит через заданную преподавателем точку плоскости с координатами , как показано на Рис. 2.

Таким образом, уравнение этой функции имеет вид В соответствии с номером варианта заданы значения параметров ограничивающей функции: . Общее количество узлов , количество попавших в область узлов (принимаем равным 39168, как ближайшим кратным 256).

Параметры моделируемой МВС:

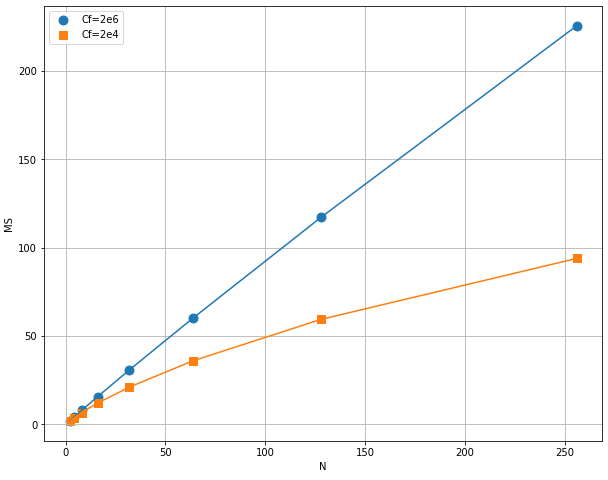
Полученные значения матожиданий и среднеквадратичных отклонений значений ускорений для целевой функции со сложностями, равномерно распределенными в интервалах , , приведены в Табл. 1, 2. Исходный код программы приведен в Приложении 1.

**Табл. 1**. Полученные результаты для различных N при

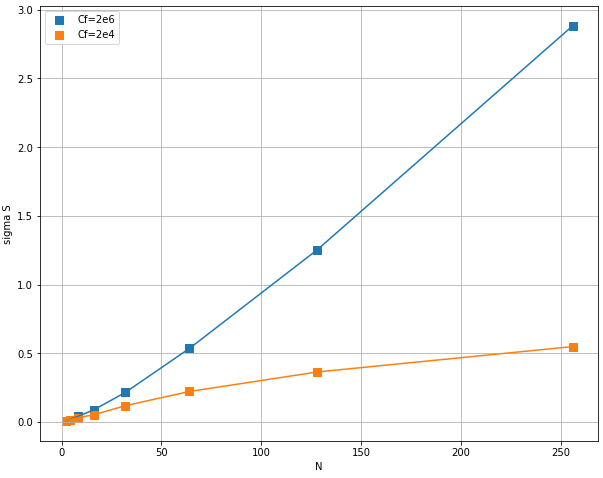
|  |  |  |
| --- | --- | --- |
| N |  |  |
| 2 | 1.92 | 2.00 |
| 4 | 3.63 | 3.96 |
| 8 | 6.68 | 7.95 |
| 16 | 12.11 | 15.81 |
| 32 | 21.23 | 31.24 |
| 64 | 36.02 | 59.70 |
| 128 | 59.99 | 119.81 |
| 256 | 93.74 | 227.59 |

**Табл. 2**. Полученные результаты для различных N при

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N |  | |  | |
|  |  |  |  |
| 2 | 1.915 | 0.007 | 1.996 | 0.009 |
| 4 | 3.613 | 0.016 | 3.974 | 0.017 |
| 8 | 6.676 | 0.031 | 7.905 | 0.040 |
| 16 | 12.076 | 0.052 | 15.672 | 0.087 |
| 32 | 21.164 | 0.119 | 30.908 | 0.216 |
| 64 | 36.103 | 0.222 | 60.371 | 0.535 |
| 128 | 59.491 | 0.364 | 117.336 | 1.253 |
| 256 | 94.003 | 0.549 | 225.764 | 2.884 |



**Рис. 3.** График



**Рис. 4.** График

**Ответы на контрольные вопросы**

1. *Чем объясняется наблюдаемый характер зависимостей ускорений , оценки математического ожидания ускорения и оценки среднего квадратического отклонения ?*

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

1. *Чем объясняется наблюдаемый характер зависимостей ускорений , оценки математического ожидания ускорения и оценки среднего квадратического отклонения от величины*

С увеличением величины 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