ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Методи аналізу швидкості паралельної обробки Інформації у GRID системах

Кваліфікаційна робота
Другий(магістерський) рівень

Автор
Костенко О.С.,
студ. гр. Спм-21-1

Керівник
Ільїна І.В.,
доц. каф. ЕОМ

Мета і задачі роботи

Мета: заповнення прогалин в аналізі та оцінці факторів, що впливають на швидкість паралельної обробки інформації у географічно розподілених інформаційних системах.

Задачі:

- проведення аналізу існуючих методів розрахунку часу та швидкості обчислень у GRID-системах
- визначення залежності швидкості обчислень у GRID-системах від кількості обчислювальних вузлів
- огляд методів оцінки оптимальної у сенсі прискорення обчислень кількості обчислювальних вузлів у GRID-системах
- розробка методик, що дозволяють проводити оцінку часу обчислень та прискорення обчислень у GRID-системах.

Актуальна класифікація GRID-систем

- обчислювальні GRID (Computational GRID);

- GRID для інтенсивної обробки даних (Data GRID);

- Семантичний GRID для оперування даними з різних баз даних (Semantic GRID).

3

Особливості паралельних обчислень у GRID-системах

Фактори, що ускладнюють здійснення паралельних обчислень у GRID-середовищі:

- нестабільність роботи великої кількості обчислювальних вузлів та мережі, через яку передаються пакети з інформацією;

- недостатня передбачуваність часу відгуку на запит

Приклади задач де доцільно використовувати GRID-технології

- масова обробка потоків експериментальних, статистичних чи інших даних великого обсягу;

- візуалізація великих наборів даних (окремі області візуального подання обробляються незалежно, а потім об'єднуються);

- складні бізнес-програми з великими обсягами обчислень.

4

Класифікація GRID-систем

За архітектурою:

- доступність;
- налаштованість;
- інтерактивність;
- керованість параметрів.

У залежності від обладнання:

- налаштованість;
- інтерактивність;
- керованість параметрів.

Стек GRID-протоколів включає 5 рівнів:

- прикладний
- колективний
- ресурсний
- зв'язуючий
- апаратний

5

Чинники, що уповільнюють процес паралельної обробкиінформації

Додаткові затримки, що виникають у GRID-системах при передачі даних, поділяються на кілька типів:

- встановлення з'єднання для початку передачі;

- передача даних для розрахунків від сервера до обчислювального вузла або результатів обчислень від обчислювального вузла до сервера;

- виникнення черги обчислювальних вузлів до сервера отримання та обробку результатів.

6

Узагальнена формула обмеження зростання продуктивності обчислювальної GRID-системи

$$S(a,c,k,t,e,r,n)=\frac{1}{\left(a+(1-a)\cdot\left(\frac{1}{c}+\frac{k}{t}+\frac{r}{e\cdot t}\cdot\left(1+p\left(z\left(c,\frac{r}{e},t\right),n\right)\cdot m\left(z\left(cm,\frac{r}{e},t\right),n\right)\right)\right)\right)}$$

де a – частка послідовних обчислень (значення в інтервалі від 0 до 1);

c – кількість обчислювальних вузлів;

k – затримка перед початком передачі (с);

t – час розрахунку одного пакета на одному обчислювальному вузлі (с);

r – обсяг пакета (біт);

e – швидкість передачі з боку сервера (біт/с);

n – кількість каналів обробки обчислювальних вузлів на сервері;

m – довжина черги до сервера (2.4);

p – ймовірність виникнення черги до сервера (2.5).

7

Узагальнена формула обмеження зростання продуктивності обчислювальної GRID-системи

Точка максимуму прискорення обчислень в залежності від кількість обчислювачів визначається за формулою: $\frac{\partial}{\partial c}S(a,c,k,t,e,r,n)=0$

Вплив кожної з тимчасових затримок показано на наступному слайді Як приклад для ілюстрації впливу тимчасових затримок були вибрані такі параметри:

- частка послідовних обчислень – 0.2;

- кількість обчислювальних вузлів варіюється;

- затримка перед початком передачі - 0.5 с;

- час розрахунку одного пакета на одному обчислювальному вузлі – 70 с;

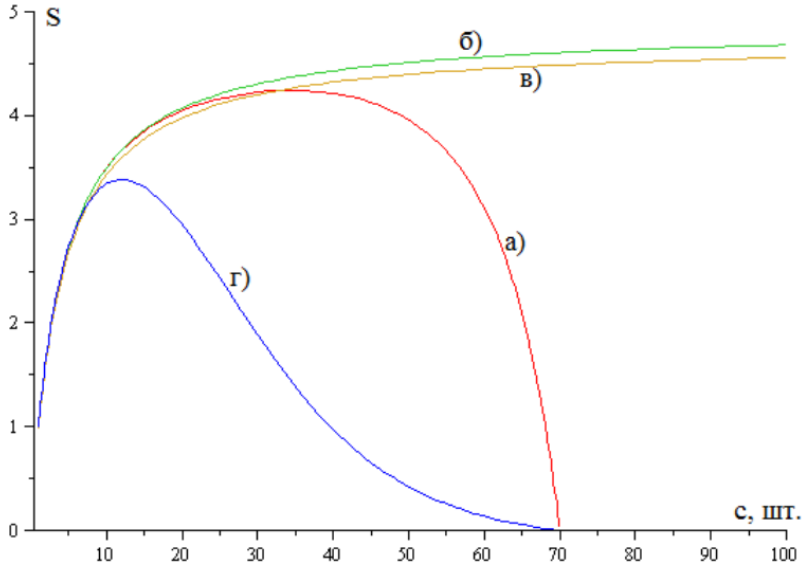
- обсяг пакету – 30 Кбіт;

- швидкість передачі з боку сервера – 30 кілобіт на секунду;

- кількість каналів обробки обчислювальних вузлів — 1.

8

Ілюстрація впливу кожної з тимчасових затримок



9

Метод оцінки швидкості паралельної обробки в GRID-системах гомогенного типу

1 $$T_A = t \cdot \left\lceil \frac{Q}{C_{max}} \right\rceil + t \cdot \begin{pmatrix} 1, \left\lceil \frac{Q}{C_{max}} \right\rceil > 0 \\ 0, \left\lceil \frac{Q}{C_{max}} \right\rceil = 0 \end{pmatrix}$$

2 $$s(0,c,t,e,r,n)^{c:\{1,..C_{max}\}} \to max$$

3 $$q \equiv \frac{r}{e \cdot t}$$

4 $$S(0,c,t,e,r,1) = \frac{1}{\frac{1}{c} + \frac{k}{t} + q \cdot (1 + p(qc,1) \cdot m(qc,1))}$$

5 $$m(qc,1) = \frac{\frac{q^2 c^2}{1 \cdot 1 (1-qc)^2}}{1 + qc + \frac{q^2 c^2}{max(1-qc,0)}}$$

6 $$m(qc,1) = \frac{q^2 c^2}{1-qc}$$

7 $$m(qc,1) = 1 - \sum_{h=0}^{1} \frac{\left(\frac{(qc)^h}{h!}\right)}{\sum_{k=0}^{1} \frac{(qc)^k}{k!} + \frac{(qc)^{1+1}}{1! \cdot (1-qc)}}$$

$$p(qc,1) = 1 - \left(\frac{1}{1 + qc + \frac{q^2 c^2}{1-qc} + 1 + qc + \frac{q^2 c^2}{1-qc}} \right) \qquad p(qc,1) = q^2 c^2.$$

8 $$S_{homegeneus}\left(0,c,t,e,r,1 = \frac{1}{\frac{1}{c} + \frac{k}{t} + q \cdot \left(1 + q^2 c^2 \cdot \frac{q^2 c^2}{1-qc}\right)}\right) \quad \frac{\partial S_{homegeneous}}{\partial c} \to max : c = \frac{t \cdot e}{2 \cdot r}$$

10 $$C_{homegeneous}(t,e,r,C_{max}) = \begin{pmatrix} 1, t < \frac{2 \cdot r}{e} \\ \frac{e}{2r} \cdot t, t \in \left[\frac{2 \cdot r}{e}, ..., \frac{2 \cdot r}{e} \cdot C_{max} \right] \\ C_{max}, t > \frac{2 \cdot r}{e} \cdot C_{max} \end{pmatrix}$$

10

Метод оцінки швидкості паралельної обробки в GRID-системах гетерогенного типу

1 $|Q_i(N)-C(N)|\to 0,$

2 $T_{stage}=t\cdot\sum_{i\in 1,j\in\overline{1,\dots,C_{all-count}}}^{Q_{all-count}}\left(\frac{Q_i}{C_j}\right)$

3 $T_{stage}=\sum_{i=1}^{Q_{all-count}}\left(t+k+\frac{r}{r}\cdot\left(1+p\left(z\left(c,\frac{r}{e},Q_i\cdot t\right),n\right)\right)\right)\cdot\left(\frac{Q_i}{C_j}\right)$ $j\in\overline{[1,\dots,C_{all-count}]},$ $|Q_i(N)-C_j(N)|\to 0,$

4 $S(0,c,t,e,r,1)=\frac{1}{\left(\frac{1}{c}+\frac{k}{t}+q\cdot(1+p(qc,1)\cdot m(qc,1))\right)\cdot\left(\frac{Q_i}{C_j}\right)}$

5 $m(qc,1)=\frac{\frac{Q^2c^2}{1\cdot 1 ctod(1-qc)^2}}{1+qc+\frac{q^2c^2}{max(1-qc,0)}}$

6 $m(qc,1)=\frac{q^2c^2}{1-qc}$

7 $p(qc,1)=1-\sum_{h=0}^{1}\frac{\left(\frac{(qc)^h}{h!}\right)}{\sum_{k=0}^{1}\frac{(qc)^k}{k!}+\frac{(qc)^{1+1}}{1!\cdot(1-qc)}};$ $p(qc,1)=1-\left(\frac{1}{1+qc+\frac{q^2c^2}{1-qc}}+\frac{qc}{1+qc+\frac{q^2c^2}{1-qc}}\right);$ $p(qc,1)=q^2c^2$

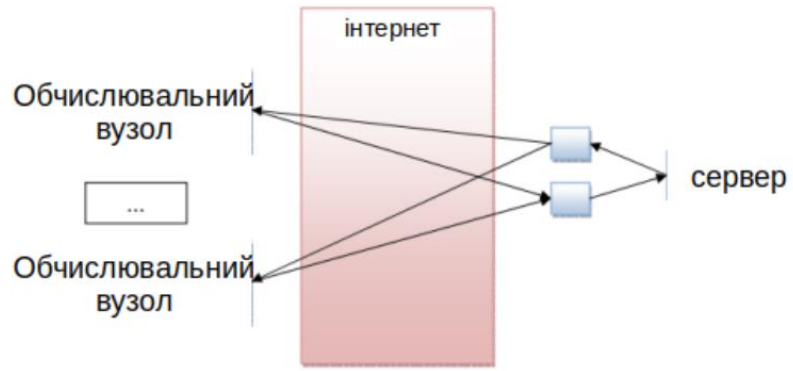
8 $S_{heterogeneous}(0,c,t,e,r,1)=\frac{1}{\sum_{i=1}^{Q_{all-count}}\left(\frac{1}{c}+\frac{k}{t}+q\cdot\left(1+q^2c^2\frac{c^2\cdot q^2}{1-qc}\right)\cdot\left(\frac{Q_i}{C_j}\right)\right)}$

$C_{heterogeneous}(t,e,r,C_{max},C(N),Q(N),Q_{all-count})=\begin{cases}1, t<\frac{2\cdot r}{e\cdot\sum_{i=1}^{Q_{all-count}}\frac{C_i}{Q_j}}\\ \frac{e\cdot\sum_{i=1}^{Q_{all-count}}\frac{C_i}{Q_j}}{2r}\cdot t, t\in\left[\frac{2\cdot r}{e\cdot\sum_{i=1}^{Q_{all-count}}\frac{C_i}{Q_j}}\cdot C_{max}\right];\\ C_{max}, t>\frac{2\cdot r}{\sum_{i=1}^{Q_{all-count}}\frac{C_i}{Q_j}}\cdot C_{max}\end{cases}$

$\frac{\partial S_{heterogeneous}}{\partial c}\to max: c=\frac{t\cdot e}{2\cdot r}\cdot\sum_{i=1}^{Q_{all-count}}\left(\frac{C_i}{Q_j}\right)$

11

Розробка архітектури та структурної схеми моделюваної GRID-системи. Імітація вирішення завдань у GRID-системі



Топологія мережі GRID-системи

12

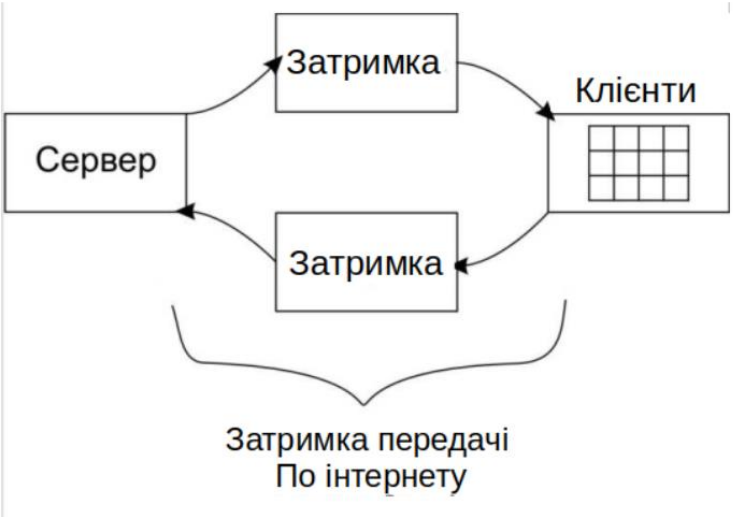
Розробка архітектури та структурної схеми моделюваної GRID-системи. Імітація вирішення завдань у GRID-системі

З МЕТОЮ ЗАБЕЗПЕЧЕННЯ МАКСИМАЛЬНОЇ НАБЛИЖЕНОСТІ УМОВ ПРОВЕДЕННЯ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ ДО РЕАЛЬНИХ, ПРИ ПРОВЕДЕННІ ЕКСПЕРИМЕНТІВ ВИКОРИСТОВУВАЛИСЯ НАСТУПНІ ВХІДНІ УМОВИ ТА ОБМЕЖЕННЯ:

- СЕРВЕР ОБРОБЛЯЄ ПАКЕТИ ВІД ОБЧИСЛЮВАЛЬНИХ ВУЗЛІВ У РЕЖИМІ ОДНІЄЇ СПІЛЬНОЇ ЧЕРГИ, ПО ЧЕРЗІ РОЗБИРАЮЧИСЬ ІЗ ЗАПИТОМ КОЖНОГО ВУЗЛА;
- ТИМЧАСОВІ ЗАТРИМКИ НА ДІЇ СЕРВЕРА, ОБЧИСЛЮВАЛЬНОГО ВУЗЛА ТА ЗАТРИМОК СЕРЕДОВИЩА ОБМІНУ ДАНИМИ ПРЕДСТАВЛЕНІ У ВИГЛЯДІ НОРМАЛЬНОГО РОЗПОДІЛУ;
- КОЖЕН ОБЧИСЛЮВАЛЬНИЙ ВУЗОЛ З ЙМОВІРНІСТЮ 0.01 МОЖЕ НА НЕВИЗНАЧЕНИЙ ЧАС ВИЙТИ З ЛАДУ, ТОБТО. ЗАТРИМАТИ РОЗРАХУНКИ ПРИЙНЯТОГО ПАКЕТА НА ВИПАДКОВИЙ ПРОМІЖОК ЧАСУ ЗА ВНУТРІШНІМ ПРИЧИН (ПЕРЕЗАВАНТАЖЕННЯ, ЗБОЇ ТОЩО).

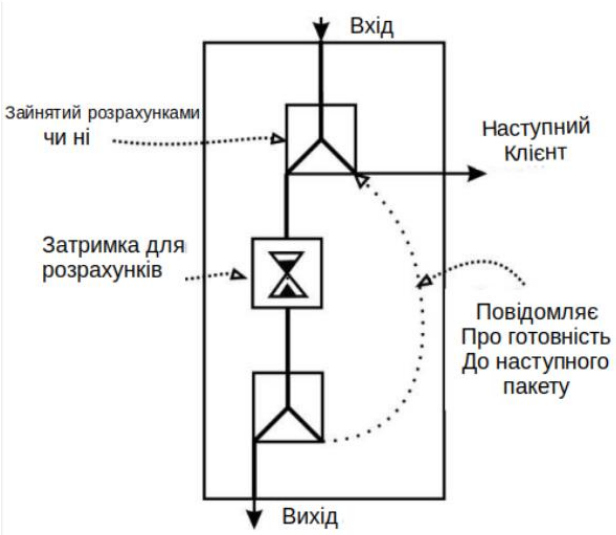
13

Узагальнена схема ГРІД-системи, що моделюється



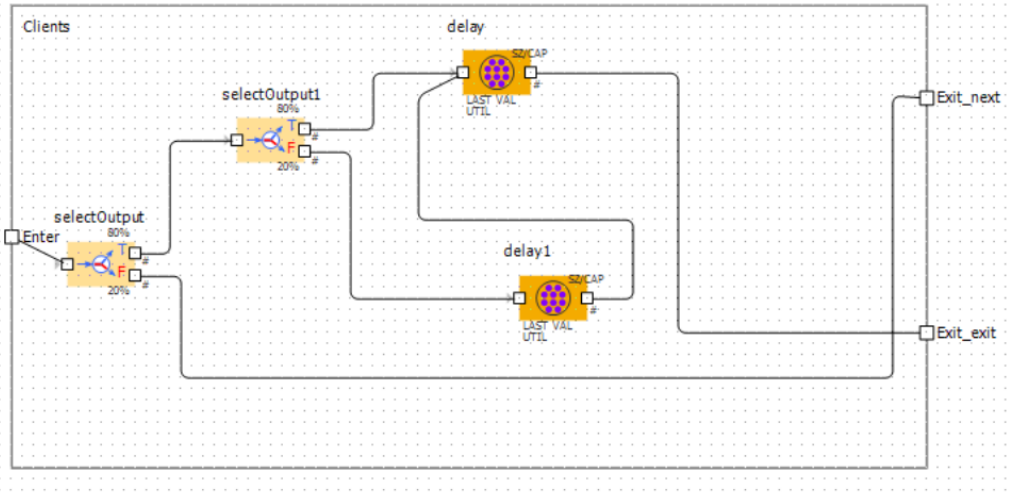
14

Структура детермінованого обчислювального вузла



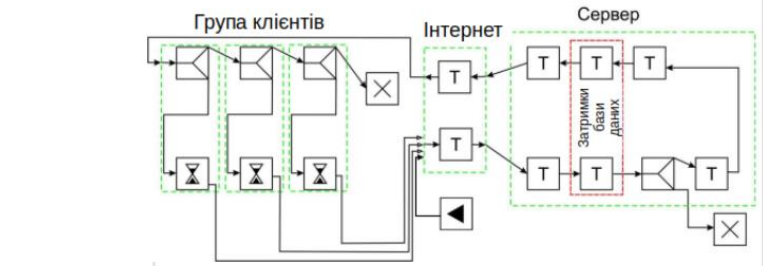
15

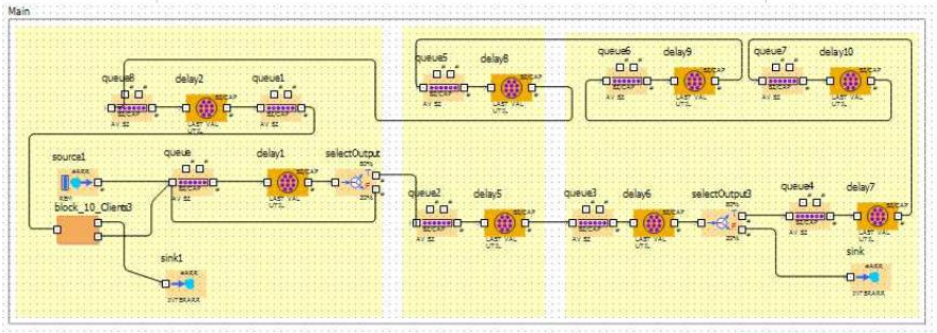
Структура обчислювального вузла з ймовірністю збільшення стандартної затримки (delay1)



16

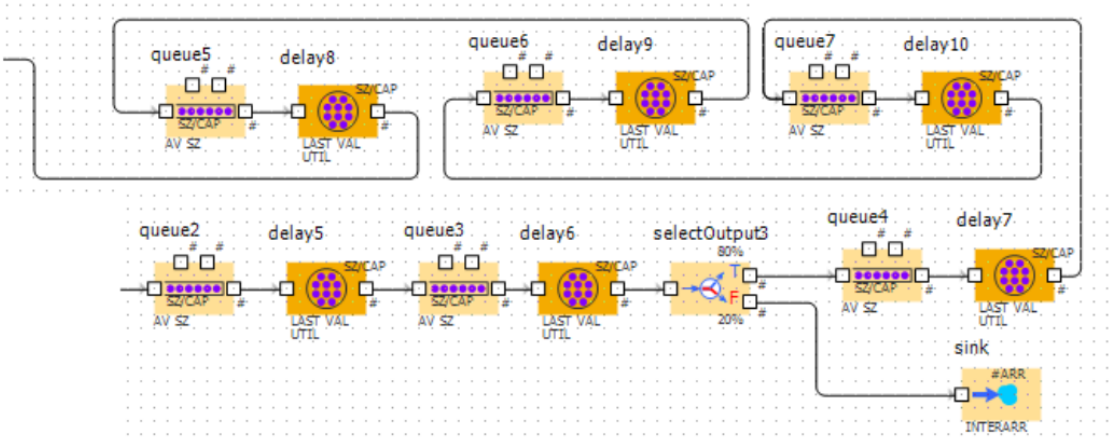
Структура GRID-системи в AnyLogic





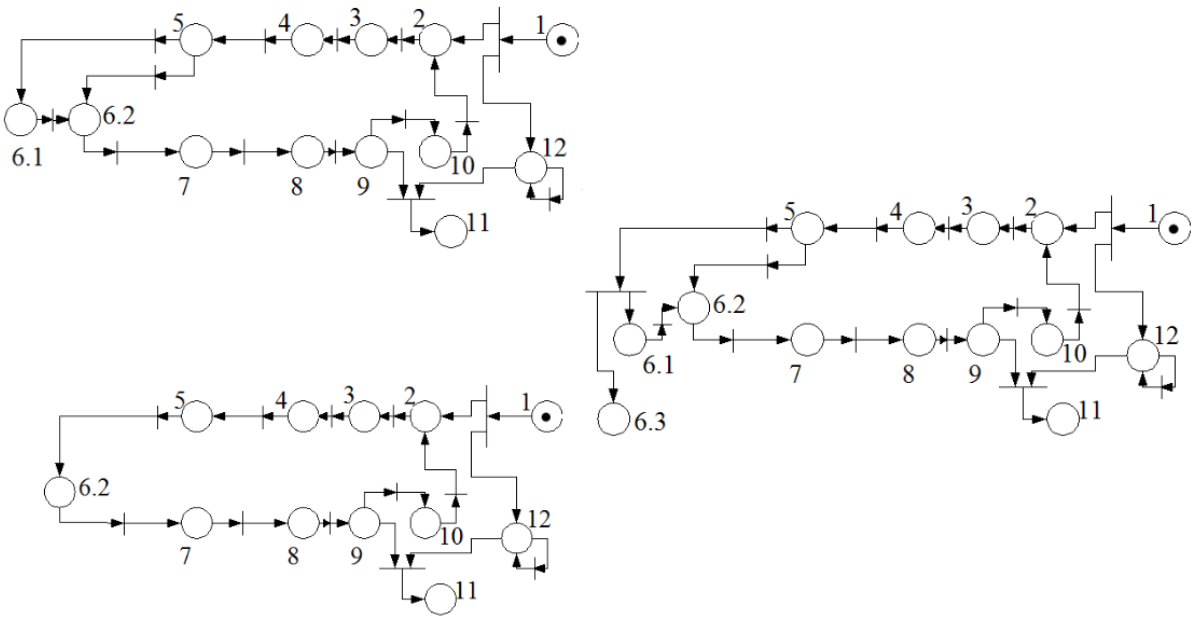
17

Блоки затримки сервера на обробку результатів та запис у БД



18

Мережі Петрі



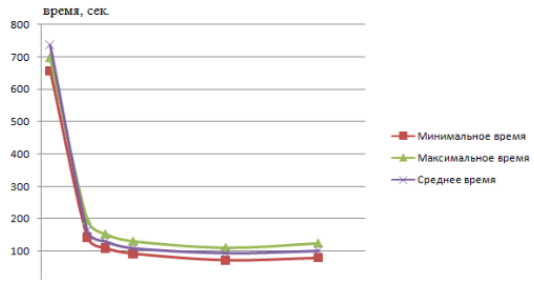
19

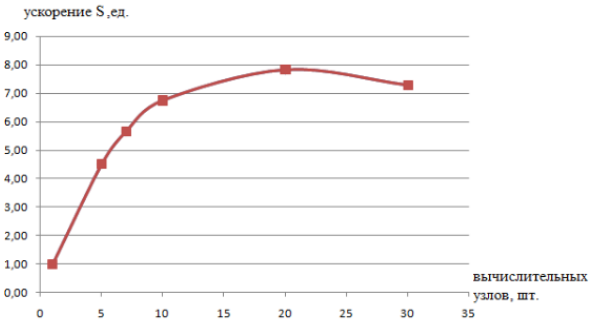
Результати експерименту

Кількість обчислювальних вузлів	Мінімальний час	Максимальний час	Середній час	Прискорення
1	687	697	737	1,00
5	144	184	163	4,53
7	144	197	130	4,23
10	100	111	109	6,78
20	83	101	94	7,87
30	86	115	101	7,27

20

Результати експерименту

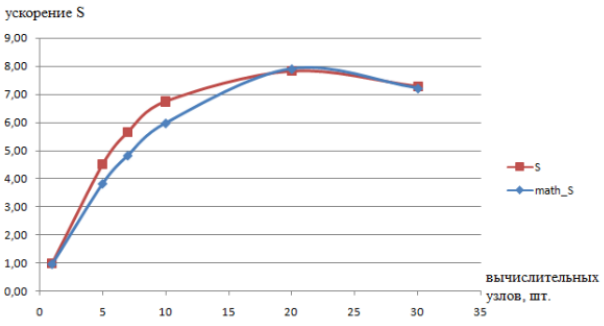




21

Значення прискорення обчислень за результатами експерименту та за формулою(7 слайд)

Кількість обчислювальних вузлів	Прискорення. [од.]	math_S/ [од,]
1	1,00	0,97
5	4,52	3,83
7	5,67	4,83
10	6,76	5,98
20	7,84	7,91
30	7,30	7,22



22

Відхилення експериментальних значень прискорення обчислень від теоретичних

Кількість обчислювальних вузлів	Прискорення [од.]	d_math_S	Відхилення значень у %
1	1,00	0,03	3,00
5	4,52	0,69	15,29
7	5,67	0,84	14,80
10	6,76	0,78	11,56
20	7,84	0,07	0,89
30	7,30	0,08	1,06

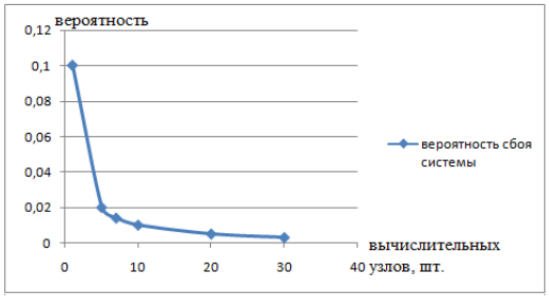
23

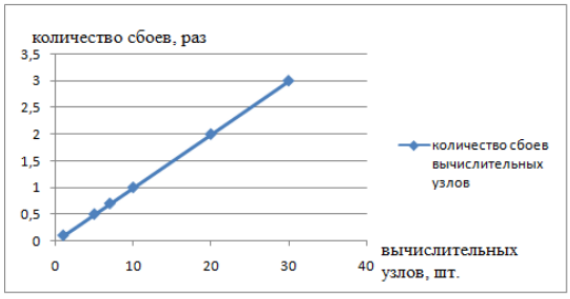
Імовірність виникнення збою в GRID-системі

Кількість пакетів	Кількість Обчислувальних вузлів	Вірогідність збою системи	Кількість збоїв обчислювальних вузлів
10	1	0,1	0,1
10	5	0,02	0,5
10	7	0,014	0,7
10	10	0,01	1
10	20	0,005	2
10	30	0,003	3

24

Імовірність виникнення збою в GRID-системі





Висновки

НА ПІДСТАВІ ВИКОНАНИХ ДОСЛІДЖЕНЬ ПРОВЕДЕНО АНАЛІЗ ПРОБЛЕМАТИКИ В ГАЛУЗІ ОРГАНІЗАЦІЇ ОБЧИСЛЕНЬ У ПАРАЛЕЛЬНИХ СИСТЕМАХ ТА СТАНУ ДОСЛІДЖЕНЬ ТА РОЗРОБОК В ОБЛАСТІ GRID-СИСТЕМ. ДОВЕДЕНО НАЯВНІСТЬ ЗАЛЕЖНОСТЕЙ МІЖ ПАРАМЕТРАМИ GRID-СИСТЕМ ТА ЧАСОМ ВИРІШЕННЯ ЗАВДАНЬ У СИСТЕМАХ, ЩО РОЗГЛЯДАЮТЬСЯ, А ТАКОЖ ДОСЯГАЄТЬСЯ ПРИСКОРЕННЯМ ОБЧИСЛЕНЬ. ВИДІЛЕНО ФАКТОРИ, ЩО УПОВІЛЬНЮЮТЬ ПРОЦЕС ПАРАЛЕЛЬНОЇ ОБРОБКИ ІНФОРМАЦІЇ. ВИКОРИСТАНО УЗАГАЛЬНЕНУ ФОРМУЛУ ОБМЕЖЕННЯ ЗРОСТАННЯ ПРОДУКТИВНОСТІ ОБЧИСЛЮВАЛЬНОЇ GRID-СИСТЕМИ. ВИЗНАЧЕНО ЗАЛЕЖНОСТІ ШВИДКОСТІ ОБЧИСЛЕНЬ У GRID-СИСТЕМАХ ВІД КІЛЬКОСТІ ОБЧИСЛЮВАЛЬНИХ ВУЗЛІВ, ЩО ДОЗВОЛЯЄ БІЛЬШ ТОЧНО ОЦІНЮВАТИ ТИМЧАСОВІ ТА РЕСУРСНІ ВИТРАТИ ДЛЯ ВИРІШЕННЯ ЗАВДАНЬ У GRID-СИСТЕМАХ. РОЗРОБЛЕНО МЕТОДИ ОЦІНКИ ОПТИМАЛЬНИХ У СЕНСІ ПРИСКОРЕННЯ ОБЧИСЛЕНЬ КІЛЬКОСТІ ОБЧИСЛЮВАЛЬНИХ ВУЗЛІВ У GRID -СИСТЕМАХ, ВРАХОВУЮЧІ ЯК ПАРАМЕТРИ GRID-СИСТЕМИ, ТАК І ПАРАМЕТРИ РОЗВ'ЯЗУВАНИХ ЗАВДАНЬ. РОЗРОБЛЕНО МЕТОДИ, ЩО ДОЗВОЛЯЮТЬ ПРОВОДИТИ АНАЛІЗ ЧАСУ ОБЧИСЛЕНЬ ТА ПРИСКОРЕННЯ ОБЧИСЛЕНЬ У GRID-СИСТЕМАХ, ЗАВДЯКИ ЧОМУ МОЖЛИВЕ БІЛЬШ ТОЧНЕ ВИЗНАЧЕННЯ НЕОБХІДНОГО РІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ ЧАСУ ЗА ВІДОМИХ ПАРАМЕТРІВ СИСТЕМИ.

ДОДАТОК Б

Програма аналізу прискорення розподілених обчислень на алгоритмах кластеризації

У цьому додатку наведено вихідний програмний код пакета " аналізатор прискорення розподілених обчислень на алгоритмах кластеризації". Мова програмування: C++.

main.cpp

```
#include <iostream>
#include <math.h>
using namespace std;
class AnalyzerInterface{
public:
virtual ~AnalyzerInterface(){}
virtual double GetAcceleration()=0;
};
class AmdahlAnalyzer: public AnalyzerInterface{
private:
//доля последовательной части задачи (0-1)
double
serial_part;
//количество вычислительных узлов
int
nodes_count;
public:
AmdahlAnalyzer(double _serial_part, int _nodes_count){
serial_part = _serial_part;
nodes_count = _nodes_count;
}
~AmdahlAnalyzer(){
}
//доля последовательной части задачи (0%-100%)
double GetSerialPart(){
return serial_part;
}
void SetSerialPart(double a){
serial_part = a;
}
//количество вычислительных узлов
int GetNodesCount(){
return nodes_count;
}
void SetNodesCount(int a){
```

```
nodes_count = a;
}
//расчет ускорения
virtual double GetAcceleration(){
double S = (double)1/( serial_part+((double)1-
serial_part)/nodes_count );
return S;
}
};
class Analyzer: public AnalyzerInterface{
private:
//доля последовательной части задачи (0-1)
double
serial_part;
//количество вычислительных узлов
int
nodes_count;
//задержка перед началом передачи данных (сек)
double
presend_delay;
//размер пакета с данными (бит)
long
pack_size;
//скорость обмена данными сервера (бит в секунду)
long
server_speed;
//время обработки пакета вычислительным узлом (сек)
double
node_calc_time;
//количество каналов сервера
int
channels_count;
//частота обращений вычислительных узлов к серверу
double
requests_frequency(int p, double s, double t){
return (s*p)/t;
}
//размер очереди к серверу
double
queue_size(double a, int n){
double top = pow(a,n+1)/(n*factorial(n)*(1-a/n)*(1-a/n));
double sum1 = 0;
for(int k=0;k<=n;++k){
sum1 += pow(a,k)/factorial(k);
}
double bottom = pow(a,n+1)/(factorial(n)*max((n-
a),(double)0.000001) );
return top/(sum1+bottom);
}
//вероятность возникновения очереди к серверу
double
queue_probability(double a, int n){
double sum1 = 0;
```

```
for(int k=0;k<=n;++k){
sum1 += pow(a,k)/factorial(k);
}
double bottom = pow(a,n+1)/(factorial(n)*max((n-
a),(double)0.000001) );
bottom += sum1;
double main_sum = 0;
for(int h=0;h<=n;++h){
main_sum += (pow(a,h)/factorial(h))/bottom;
}
main_sum = 1-main_sum;
return min(main_sum,(double)1);
}
//функция расчета факториала
long factorial(int a){
if(a==0){
return 1;
}
long result = 1;
for(int i=a;i>1;--i){
result *= i;
}
return result;
}
public:
Analyzer(double _serial_part,
int _nodes_count,
double _presend_delay,
long _pack_size,
long _server_speed,
double _node_calc_time,
int _channels_count){
serial_part = _serial_part;
nodes_count = _nodes_count;
presend_delay = _presend_delay;
pack_size = _pack_size;
server_speed = _server_speed;
node_calc_time = _node_calc_time;
channels_count = _channels_count;
}
~Analyzer(){
}
//доля последовательной части задачи (0%-100%)
double GetSerialPart(){
return serial_part;
}
void SetSerialPart(double a){
serial_part = a;
}
//количество вычислительных узлов
int GetNodesCount(){
return nodes_count;
}
```

```
void SetNodesCount(int a){
nodes_count = a;
}
//задержка перед началом передачи данных (сек)
double GetPresendDelay(){
return presend_delay;
}
void SetPresendDelay(double a){
presend_delay = a;
}
//размер пакета с данными (бит)
long GetPackSize(){
return pack_size;
}
void SetPackSize(long a){
pack_size = a;
}
//скорость обмена данными сервера (бит в секунду)
long GetServerSpeed(){
return server_speed;
}
void SetServerSpeed(long a){
server_speed = a;
}
//время обработки пакета вычислительным узлом (сек)
double GetNodeCalcTime(){
return node_calc_time;
}
void SetNodeCalcTime(double a){
node_calc_time = a;
}
//количество каналов сервера
int GetChannelsCount(){
return channels_count;
}
void SetChannelsCount(int a){
channels_count = a;
}
//расчет ускорения
virtual double GetAcceleration(){
double z = requests_frequency(
nodes_count,
(double)pack_size/(double)server_speed,
node_calc_time);
double p1 =
queue_probability(z,channels_count)*queue_size(z,channels_count)
+1;
double p2 = (double)1/(double)nodes_count
+(double)presend_delay/(double)node_calc_time
+(double)pack_size*p1/(double)(node_calc_time*server_speed);
double S = (double)1/( serial_part+((double)1-
serial_part)*p2 );
return S;
```

```
}
};
int main()
{
cout<<"Serial part (0-1): ";
double a;
cin>>a;
cout<<"Calculating nodes: ";
int b;
cin>>b;
cout<<"Delay before data transaction (sec): ";
double c;
cin>>c;
cout<<"Data package size (bit): ";
long d;
cin>>d;
cout<<"Server data exchange speed (bit/sec): ";
long e;
cin>>e;
cout<<"Node calculation time (sec): ";
double f;
cin>>f;
cout<<"Server channels count: ";
int g;
cin>>g;
AmdahlAnalyzer *am_analyzer = new AmdahlAnalyzer(a,b);
Analyzer *analyzer = new Analyzer(a,b,c,d,e,f,g);
printf("amdahl:%.2f \nnew:%.2f\n",am_analyzer-
>GetAcceleration(),
analyzer->GetAcceleration());
delete analyzer;
delete am_analyzer;
return 0;
}
```