SECTION 2. Applied mathematics. Mathematical modeling.

Shevtsov Alexandr Nikolayevich,
candidate of technical Sciences,
associateProfessor of the Department
«Applied mathematics»,
Taraz State University named after
M.Kh. Dulati, Kazakhstan

Beken Berik Kadyrbergenovich
1st year student of the speciality
"Mathematics "
Taraz State University named after M.Kh.
Dulati, Kazakhstan

Talasbayev Adilgazy Amangaliyevich
1st year student of the speciality
"Mathematics "
Taraz State University named after
M.Kh. Dulati, Kazakhstan

STUDY OF PARALLEL COMPUTATIONS ON DELPHI

The paper considers the methods and algorithms for the implementation of multithreaded calculations on Delphi for modern multi-core computers. Made comparison of algorithms for simple examples.

Keywords: parallel computing, multi-core computer, Delphi, streams.

УДК 004.021

ИССЛЕДОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА DELPHI

В работе рассмотрены методы и алгоритмы реализации многопоточных расчетов на Дельфи для современных многоядерных компьютеров. Сделано сравнение алгоритмов на простых примерах.

Ключевые слова: параллельные вычисления, многоядерный компьютер, Дельфи, потоки.

В последние годы усиливается внимание к использованию высокопроизво-дительной вычислительной техники. Суперкомпьютеры разрабатываются в первую очередь для того, чтобы с их помощью решать сложные задачи, требующие огромных объемов вычислений. При этом подразумевается, что может быть создана единая программа, для выполнения которой будут задействованы все ресурсы суперкомпьютера. Однако не всегда такая единая программа может быть создана или ее создание целесообразно. В самом деле, при разработке программы для многопроцессорной системы мало разбить программу на параллельные ветви. Для эффективного использования ресурсов необходимо обеспечить равномерную загрузку всех процессоров, что в свою очередь означает, что все ветви программы должны выполнить примерно одинаковый объем вычислительной работы. Однако не всегда этого можно достичь.

Delphi представляет программисту полный доступ к возможностям программирования интерфейса Win32 и Win64 [1]. Для организации потоков предоставляется специальный класс TThread. Класс полностью упрощает программный интерфейс, предоставляя разработчику простой доступ к программированию потоков. В двух словах, все, что нам необходимо сделать, — это перекрыть виртуальный метод Execute[2-4].

Однако при попытке практической реализации можно столкнуться с целым рядом проблем, связанных со сложностью выбора потоков, правильным разбиением кода, и оценкой необходимости самих многопоточных вычислений[5-6].

Разработаем программу выполняющую определенный объем вычислительной работы, точнее, расчет сумм корней из последовательно увеличивающихся индексов (Рис.1.).

$$k_1 = \sum_{1}^{100\,000\,000} \sqrt{i},$$

$$k_2 = \sum_{1}^{100\,000\,000} \sqrt{i}.$$

Рассмотрим вначале реализацию этого рассчета в одном последовательном потоке, а затем при создании второго дополнительного потока.

code: Delphi

```
procedure TForm1.Button1Click(Sender: TObject);
var t : longint;
```

```
    I: Integer;
begin
t := GetTickCount;                    // Начало
k1:=0;              k2:=0;
memo1.Clear;     memo2.Clear;

for I := 1 to 100 000 000 do          k1:=k1+sqrt(i);
for I := 1 to 100 000 000 do          k2:=k2+sqrt(i);

memo1.Text:=floattostr(k1);          memo2.Text:=floattostr(k2);

t := t - GetTickCount ;
statusbar1.Panels.Items[1].Text:= 'Время работы: '+floattostr(t/1000)+ ' сек.';
label1.Caption:=statusbar1.Panels.Items[1].Text;
end;
```

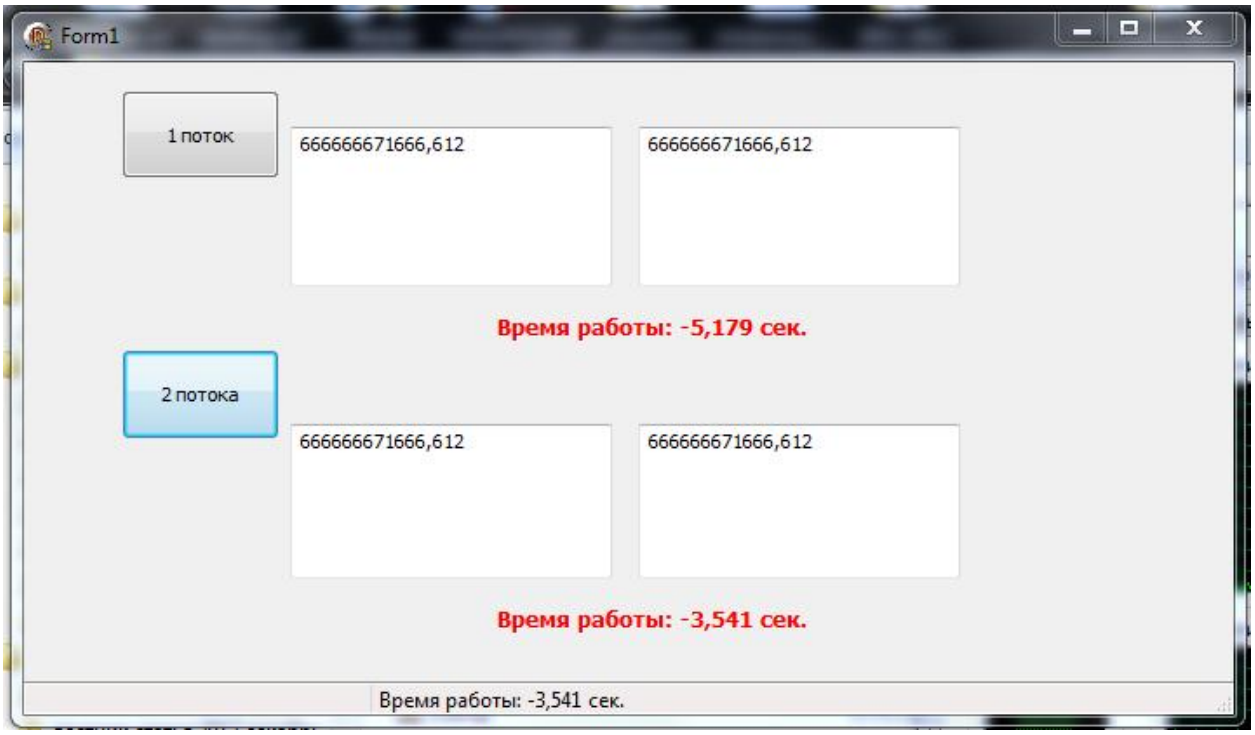

Рисунок 1- Окно программы.

Алгоритм создания второго потока зададим в виде:

code: Delphi

```
  procedure TPotok2.Execute;
var i:integer;
begin
```

```
for I := 1 to 100000000 do
k2:=k2+sqrt(i);
end;

procedure TForm1.Button2Click(Sender: TObject);
var t : longint;
  I: Integer;
begin
t := GetTickCount; // Начало
k1:=0;
k2:=0;
memo3.Clear;
memo4.Clear;

for I := 1 to 100000000 do
k1:=k1+sqrt(i);

Potok2.Resume;
while not(potok2.Finished) do ;

memo3.Text:=floattostr(k1);
memo4.Text:=floattostr(k2);

t := t - GetTickCount ;
statusbar1.Panels.Items[1].Text:= 'Время работы: '+floattostr(t/1000)+ ' сек.';
 label2.Caption:=statusbar1.Panels.Items[1].Text;
end;
```

При работе программы, будем анализировать время работы каждого алгоритма и загрузку ядер центрального процессора (Рис.2.):

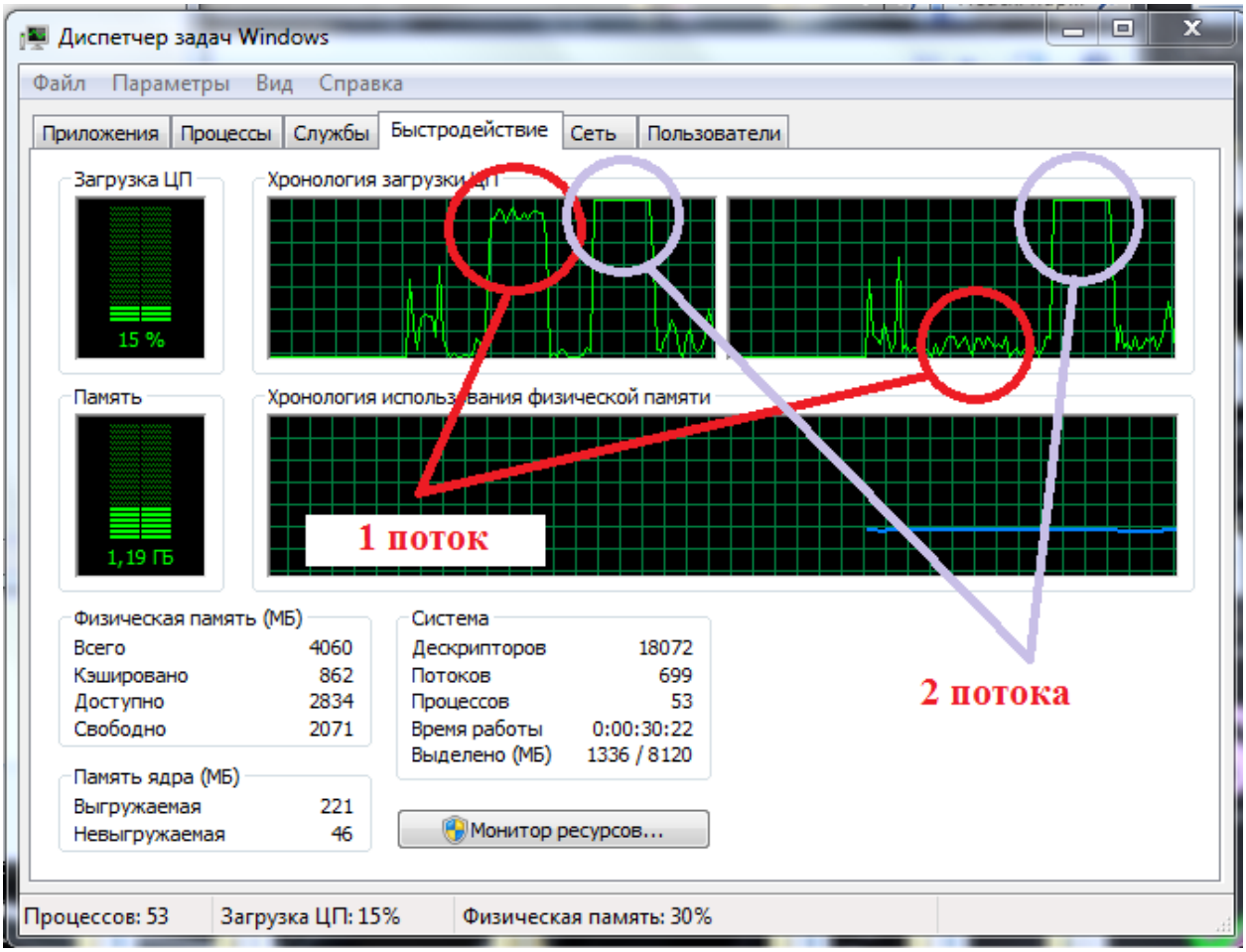

Рисунок 2 - Загрузка ЦП при одном и двух потоках.

Получаем уменьшение времени расчета алгоритма около полторы секунды при двух поточном вычислении.

Рассмотрим другие вычислительные процессы и оценим эффективность их выполнения при 1 и двух потоках. Изменим формулу на сумму кубических корней:

$$k_1 = \sum_{1}^{100\,000\,000} \sqrt[3]{i},$$

$$k_2 = \sum_{1}^{100\,000\,000} \sqrt[3]{i}.$$

Тогда имеем (Рис.3,4), уменьшение времени рассчета с 25 секунд, до 15 секунд. Алгоритм в 2 потоках почти в 2 раза эффективнее чем в один.

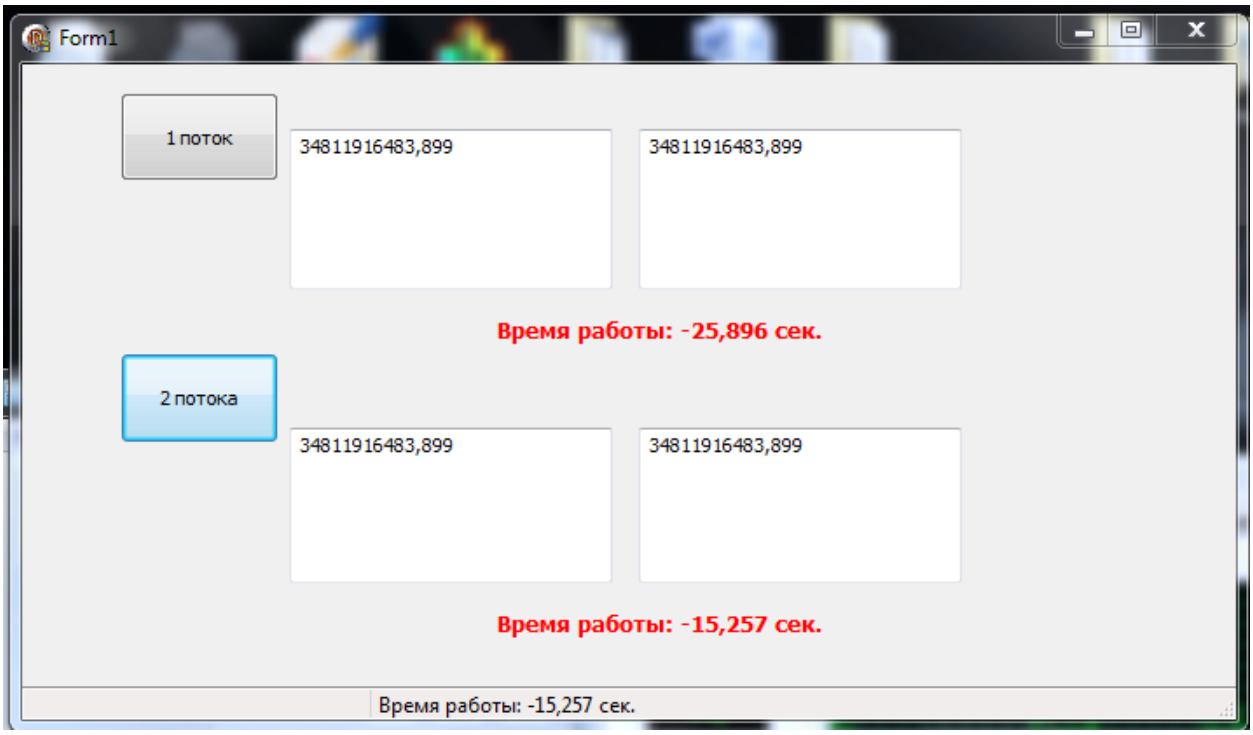

Рисунок 3 - Расчет кубических корней.

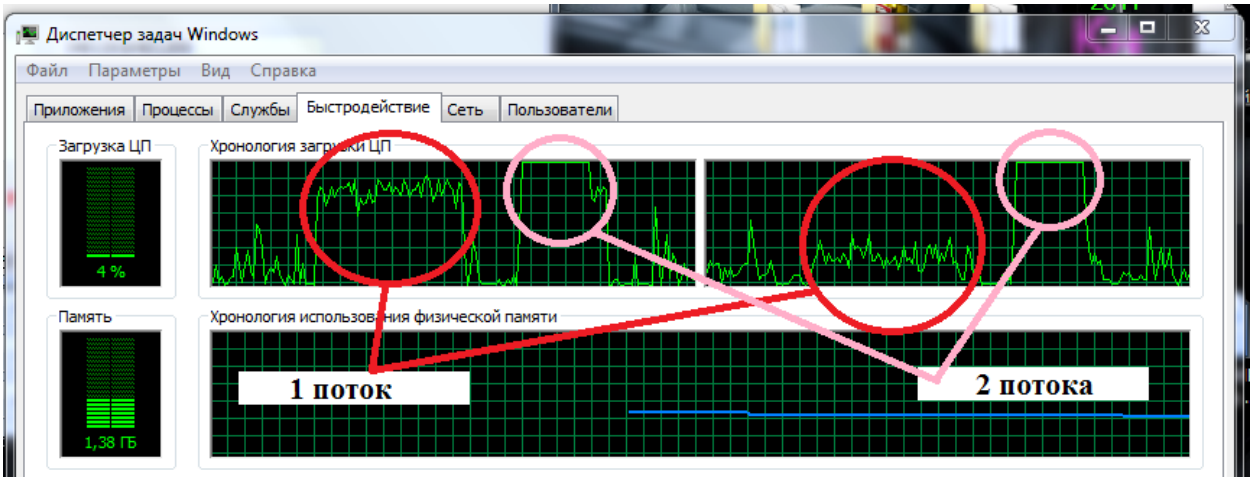

Рисунок 4 - Загруженность ядер процессора при расчете кубических корней.

Сделаем расчет, для квадратов индексов (Рис.5,6):

$$k_1 = \sum_{1}^{100\,000\,000} i^2,$$

$$k_2 = \sum_{1}^{100\,000\,000} i^2.$$

Имеем задержку в расчете почти на 0,3 секунды. Это можно объяснить тем – что при небольших расчетах (которые занимают небольшое процессорное время) использование многопоточных

вычислений – нецелесообразно. В данном случае мы и получаем – почти одинаковое время выполнения алгоритмов – 2 секунды. В случае же (Рис.3) сложных вычислений, мы получаем очень большой прирост производительности алгоритма – в 10 секунд.

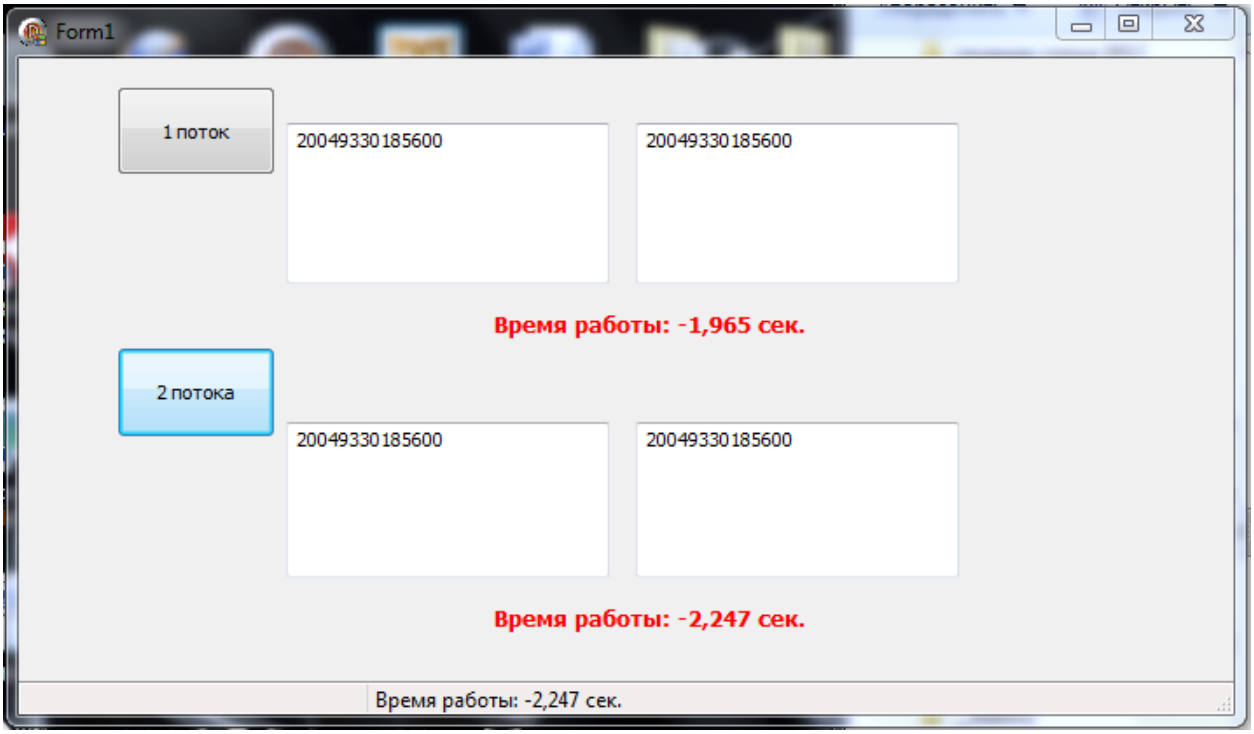

Рисунок 5 - Расчет суммы квадратов.

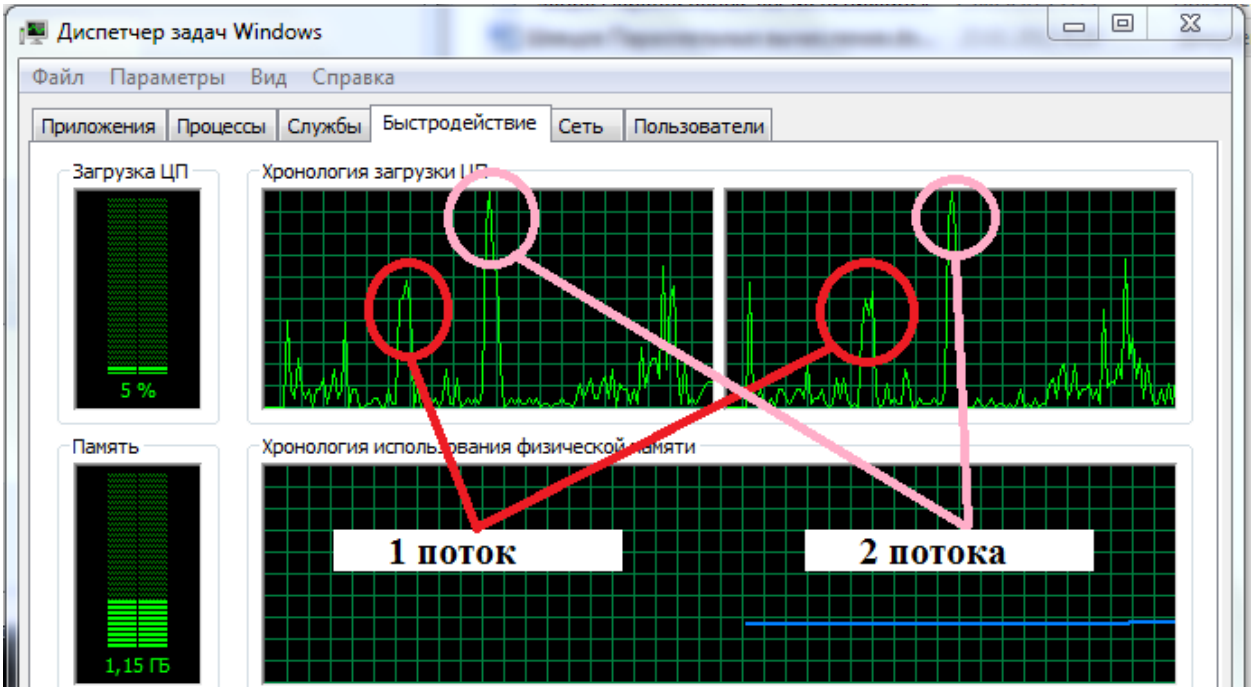

Рисунок 6 - Загруженность ядер процессора при расчете суммы квадратов.

Рассмотрим теперь сумму синусов:

$$k_1 = \sum_{1}^{100000000} \sin i,$$

$$k_2 = \sum_{1}^{100000000} \sin i.$$

Получим на 4 секунды увеличение быстродействия алгоритма в двух потоках (Рис.7,8).

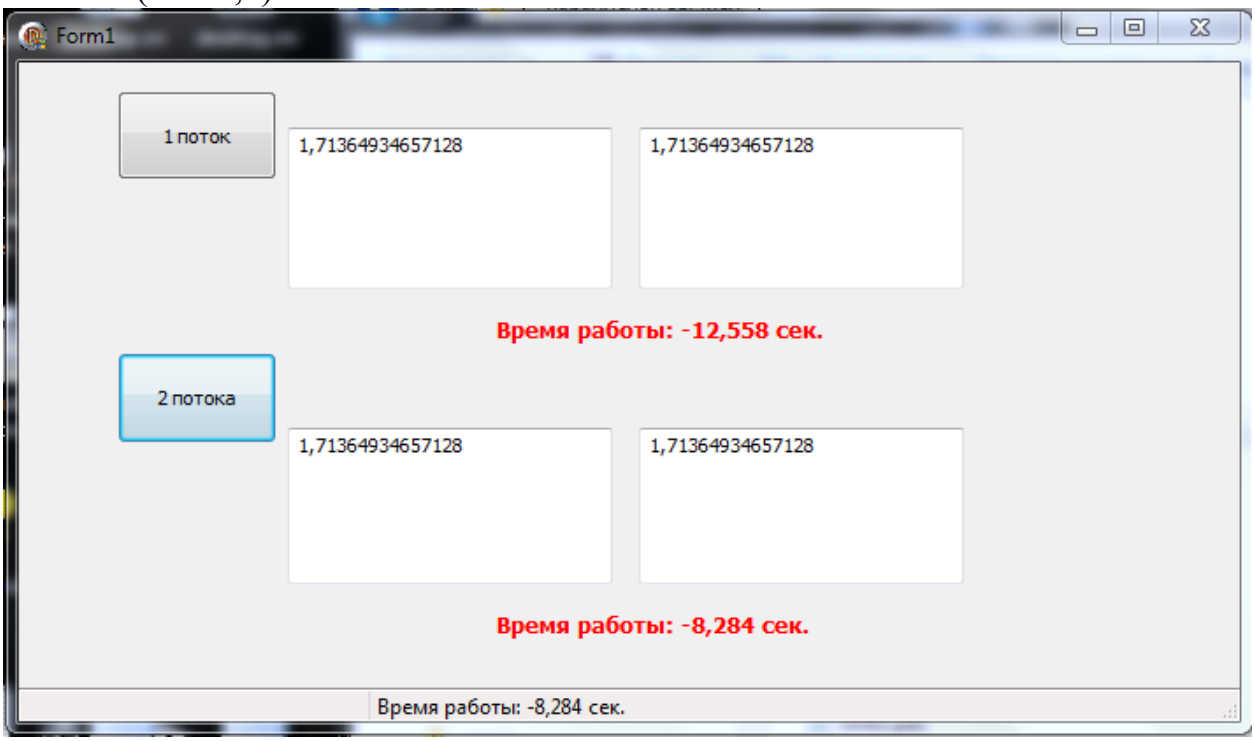

Рисунок 7 - Расчет сумм синусов.

Следует отметить небольшой спад загруженности при выполнении расчета в 2-х потоках. Это объясняется тем, что один из потоков уже завершил свою работу а второй еще продолжал выполнение. Также при выполнении всех алгоритмов была замечена – работа обоих ядер. Это показывает наличие встроенной в операционную систему «Windows 7 Максимальная» системы распараллеливания отдельных расчетов. Наиболее приспособленными для этого оказалось вычисление квадратов чисел и синусов, а наименее – вычисление корней, экспонент и логарифмов.

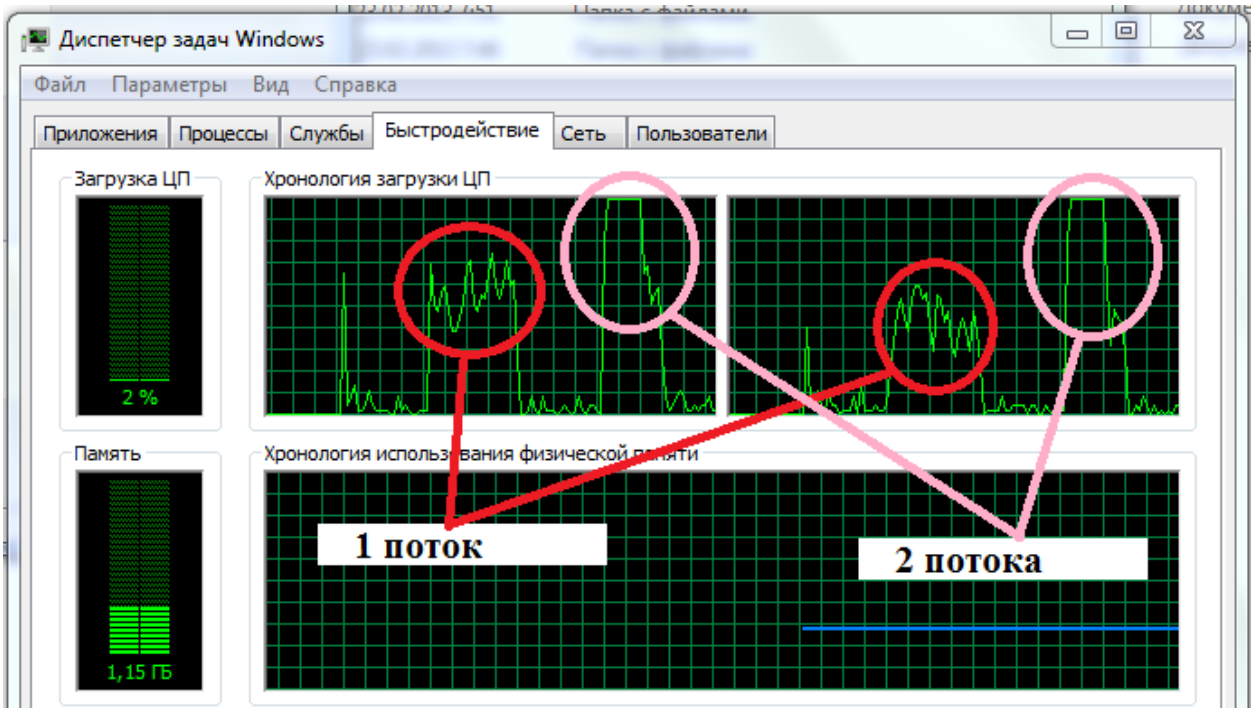

Рисунок 8 - Загруженность ядер процессора при расчете суммы синусов.

Литература.

1. TThread в Delphi. /Учебник по Delphi для профессионалов/ [Электронный ресурс]. URL:**http://rusdir.blogspot.com/2010/02/tthread-delphi.html** (дата обращения: 18.03.2013).
2. Антонов А.С. Введение в параллельные вычисления.-Методическое пособие. -Москва, 2002г. -69с.
3. Создание потоков средствами класса Tthread [Электронный ресурс]. URL:**http://www.codingrus.ru/readarticle.php?article_id=1999** (дата обращения: 12.03.2013).
4. Первые шаги с TThread в Delphi[Электронный ресурс]. URL:**http://www.codingrus.ru/readarticle.php?article_id=1999** (дата обращения: 22.04.2013).
5. Аппаратные Технологии Многоядерных Процессоров[Электронный ресурс]. URL:**http://www.fistpgtu.ru/index.php?option=com_content&task=view&id=213&Itemid=2** (дата обращения: 22.04.2013).
6. Заплавный А.Г. Проблема эффективности применения хэш-функции MD5 с учетом современных вычислительных возможностей и параллельных вычислений. [Электронный ресурс]. URL:**http://stavkombez.ru/conf/category/section7/** (дата обращения: 22.04.2013).