ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ КАЗАХСТАН
MINISTRY OF EDUCATION AND SCIENCE
OF THE REPUBLIC OF KAZAKHSTAN

Л.Н. ГУМИЛЕВ АТЫНДАҒЫ
ЕУРАЗИЯ ҰЛТТЫҚ УНИВЕРСИТЕТІ
ЕВРАЗИЙСКИЙ НАЦИОНАЛЬНЫЙ
УНИВЕРСИТЕТ ИМ. Л.Н. ГУМИЛЕВА
GUMILYOV EURASIAN
NATIONAL UNIVERSITY

ЖАС ҒАЛЫМДАР КЕҢЕСІ

Студенттер мен жас ғалымдардың «Ғылым және білім - 2015» атты X Халықаралық ғылыми конференциясының БАЯНДАМАЛАР ЖИНАҒЫ

СБОРНИК МАТЕРИАЛОВ X Международной научной конференции студентов и молодых ученых «Наука и образование - 2015»

PROCEEDINGS of the X International Scientific Conference for students and young scholars «Science and education - 2015»

УДК 001:37.0
ББК72+74.04
Ғ 96

Ғ96
«Ғылым және білім – 2015» атты студенттер мен жас ғалымдардың X Халық. ғыл. конф. = X Межд. науч. конф. студентов и молодых ученых «Наука и образование - 2015» = The X International Scientific Conference for students and young scholars «Science and education - 2015». – Астана: http://www.enu.kz/ru/nauka/nauka-i-obrazovanie-2015/, 2015. – 7419 стр. қазақша, орысша, ағылшынша.

ISBN 978-9965-31-695-1

Жинаққа студенттердің, магистранттардың, докоранттардың және жас ғалымдардың жаратылыстану-техникалық және гуманитарлық ғылымдардың өзекті мәселелері бойынша баяндамалары енгізілген.

The proceedings are the papers of students, undergraduates, doctoral students and young researchers on topical issues of natural and technical sciences and humanities.

В сборник вошли доклады студентов, магистрантов, докторантов и молодых ученых по актуальным вопросам естественно-технических и гуманитарных наук.

УДК 001:37.0
ББК 72+74.04

ISBN 978-9965-31-695-1

ӘОК 004.4/.5

MICROSOFT VISUAL C# ОРТАСЫНДА ПАРАЛЛЕЛЬ ЕСЕПТЕУЛЕРДІҢ ҚОЛДАНЫЛАТЫН ӘДІСТЕРІ

Калибекова Данара Шайкуалиевна
Be_attentive@mail.ru
Қазақстан, Астана, Л.Н.Гумилев атындағы ЕҰУ
Информатика кафедрасының оқытушысы

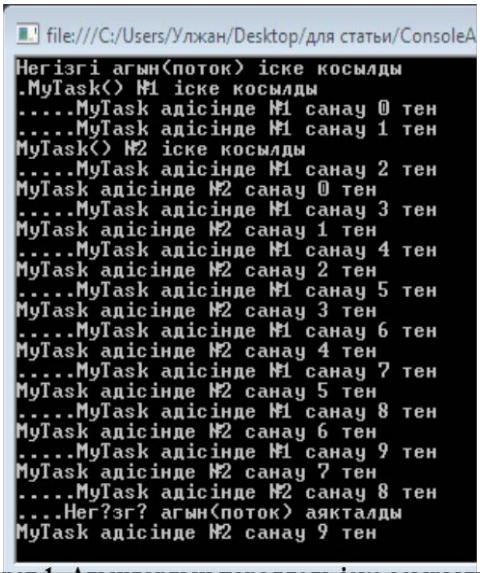
Бірнеше үрдістерді бірден орындау мүмкіндігі бар көп ядролы машиналардың дәуірінде .Net ағымымен стандартты құрал жұмыстарын жасау жеткіліксіз. Сондықтан .Net 4.0 платформасының жаңалығы негізгі қызметі System.Threading.Tasks атаулар кеңістігінде орналасқан параллельді есептеу TPL(Task Parallel Library)кітапханасы болып табылады. Аталған кітапхана тапсырмаларды параллельдеуге және егерде компьютер бірнеше ядродан тұратын болса, бірнеше процессорда бірден орындауға мүмкіндік береді.

Параллель есептеулер- бір уақытта жұмыс істейтін өзара әрекеттесуші есептеу үрдістерінің жиынытығы ретінде бағдарламалар өңделетін компьютерлік есептеулердің ұйымдастырылу тәсілдері[1].

Параллель есептеулердің тарихы тізбекті есептеу концепциясы туындағаннан бастау алды. Алғашқы параллель есептеудің тұтас моделі ретінде Дж.Фон Нейманның торлы автоматтар концепциясын жатқызуға болады. Параллель есептеулердің алғашқы теориялық зерттеулерін Карп және Миллер өз жұмыстарында қарастырды. Параллель есептеулерді зерттеу барысында көптеген тізбекті есептеулерге тән емес қасиеттер мен мәселелерді анықтады. Сонымен қатар, параллель алгоритмдеудің алғашқы қадамдары анықталды: ауқымды және «болжамсыз», ағынды(data-flow) және есептеуді жүзеге асырудың операторлы сызбалары. Заманауи зерттеулер мен жұмыстар көбінесе көп ядролы процессорлар мен көп процессорлы жүйелерді(Intel, AMD, IBM) өңдеуге шоғырланған. Параллель есептеу жүйелерін қолдану есептеу техникасының стратегиялық бағыттағы дамуы болып табылады. Параллельді есептеулердің дамуы тек қана бір уақытта бірнеше амалдарды орындаумен ғана шектелмейді, сонымен қатар көп ядролы машиналардың дамуымен байланысты[2].

Microsoft Visual Studio бағдарламасының негізінде параллель есептеудің қарапайым мысалын қарастырайық. Төменде берілген мысалда екі есеп құрылады да қайсысы орындалатынын көрсетеді:

```
ClassProgram
        {
StaticvoidMyTask ()
      {
Console.WriteLine ("My Task () № {0} іске қосылды",
Task.CurrentId);
for (int count = 0; count < 10; count++)
        {
Thread.Sleep(500);
Console.WriteLine ("MyTask әдісінде № {0} санау {1} тең ",
Task.CurrentId, count);}}
Staticvoid Main (string [] args)
      {   Console.WriteLine ("Негізгіағын (поток) іске қосылды");
Task task1 = newTask(MyTask);
Task task2 = newTask(MyTask);
// Есепті іске қосу
task1.Start ();task2.Start ();
for (inti = 0; i< 60; i++)
{Console. Write (".");
Thread. Sleep (100);          }
Console.WriteLine ("Негізгіағын(поток) аяқталды");
Console.ReadLine();} [3]
```



Сурет 1- Ағындардың параллель іске асырылуы

Есептің аяқталуын күтуді ұйымдастыру үшін Task класындағы арнайы ұсынылатын тәсілдерді де қолдануға болады. Бұл класстағы ең қарапайым әдіс Wait() болып табылады. Аталған әдісті қолдану кезінде екі ерекше жағдай туындауы мүмкін. Біріншісі *ObjectDisposedException* ерекше жағдайы болып табылады. Бұл жағдай Dispose() әдісімен тікелей босатылған жағдайда орындалады. Ал, екінші *AggregateException* ерекше жағдайы есеп өздігінен тоқтаған кезде орындалады. Wait() әдісін әрбір жағдайға байланысты шақыру барысында командаларды біріктіре отырып, **WaitAll()** әдісін қолдануға болады. Бұл әдіс есептердің тобын күтуді ұйымдастырады. Аяқталуын күтуді талап ететін есептер tasks массиві түріндегі параметрлердің көмегімен тасымалданады. Бұл жағдайда әртүрлі ерекше жағдайлар туындауы мүмкін:

```
/* task1.Wait();
task2.Wait();*/

Task.WaitAll(task1, task2);
```

Кейбір жағдайда топтастырылған тапсырмалардың кез келгенін тоқтатылуын күту қажет болады. Сондай жағдайда WaitAny() әдісі қызмет атқарады. Бұл әдісті сипаттаудың қарапайым формасын келтірейік:

public static int WaitAny(params Task[] tasks)

Task.WaitAny(task1, task2);

Параллельді есептеулер барысында қолданылатын әдістер:

- Dispose әдісі. Бұл әдіс Task класында жүзеге асырылады. Бағдарлама аяқталу барысында автоматты түрде босатылады. Сипатталуы:

public void Dispose()

- TaskFactory класы. Кез келген есептерді қарастыру барысында бірден есептеуге көшетіндей тиімді әдістер қолдануға болады. Бұл үрдісті жүзеге асыру үшін StartNew() әдісі қолданылады. Сипатталуы:

public Task StartNew(Action action)

мұндағы, action-орындалатын тапсырмаға ену нүктесі.

- TPL кітапханасының ең жаңашыл және тиімді ерекшеліктерінің бірі тапсырманың жалғасын құра алу мүмкіндігі. Жалғастыру- бір тапсырма аяқталғаннан кейін, автоматты түрде келесі бір тапсыманың басталуы. Сонымен қатар, тапсырмалардың жалғасуын ContinueWith() әдісінің көмегімен де жүргізуге болады. Сипатталуы:

public Task ContinueWith(Action<Task>жалғасу қызметі)

- Invoke() әдісі. Аргументтер түрінде берілген бірнеше әдістерді орындауға мүмкіндік береді. Қолжетімді процессорларды қолданыла отырып, кодтың орындалуын кеңейтеді. Сипатталуы:

public static void Invoke(params Action[] actions)

- For әдісі. Циклдың әрбір қадамында циклды басқару айнымалысы бір бірлікке ұлғаяды. Сипатталуы:

public static ParallelLoopResult

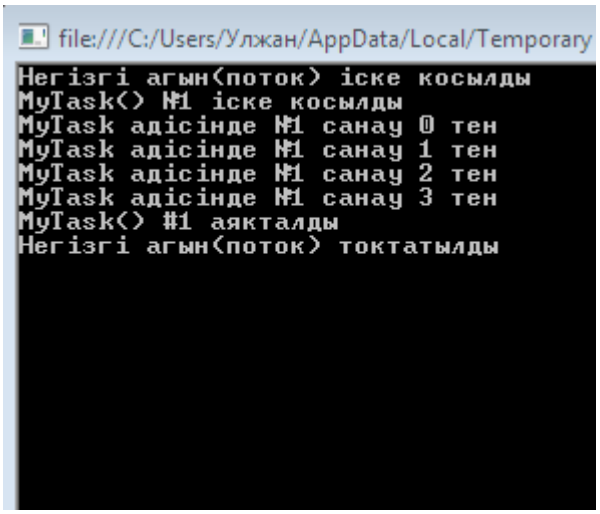
For(int fromInclusive, int toExclusive, Action<int> body)

мұндағы, fromInclusive- циклды басқару айнымалысының бастапқы мәтіне сәйкес келеді; toExclusive- соңғы мәннен бір бірлікке жоғары мәні. For әдісінің ең негізгі ерекшелігі циклда кодтың орындалуын параллельдеуге мүмкіндік береді.

Жоғарыда қарастырылған әдістерге сәйкес мысал қарастырайық:

```
using System.Threading;
using System.Threading.Tasks;
namespace ConsoleApplication1
{    class Program
  {      static void MyTask(Object ct)
{ CancellationToken cancelTok = (CancellationToken)ct;
```

```
        cancelTok.ThrowIfCancellationRequested();
Console.WriteLine("MyTask() №{0} іске қосылды",Task.CurrentId);
for (int count = 0; count < 10; count++)
{// қолданылатын сұраныс
if (!cancelTok.IsCancellationRequested)
{Thread.Sleep(500);
Console.WriteLine(" MyTask әдісінде №{0} санау {1} тең", Task.CurrentId, count);}}
Console.WriteLine("MyTask() #{0} аяқталды",Task.CurrentId);  }
      static void Main()
      {  Console.WriteLine("Негізгі ағын(поток) іске қосылды");
         // болдырмау белгілерінің объектілері
         CancellationTokenSource cancelTokSSrc =
new CancellationTokenSource();
         // Болдырмау белгісін тасымалдай отырып,
есепті іске қосу
Task tsk = Task.Factory.StartNew(MyTask,
cancelTokSSrc.Token, cancelTokSSrc.Token);
         Thread.Sleep(2000);
         Try  {
            // есепті болдырмау
            cancelTokSSrc.Cancel();  tsk.Wait();}
         catch (AggregateException exc)
         {   if (tsk.IsCanceled)
Console.WriteLine("tsk  тапсырмасы тоқтатылды");
         }   finally        {
            tsk.Dispose();  cancelTokSSrc.Dispose();  }
         Console.WriteLine("Негізгі ағын(поток) аяқталды");
         Console.ReadLine();  }   }}[3]
```



Сурет 2- Болдырмау әдісін қолдану

Параллель есептеулер сіздің бағдарламалық өніміңіздің жылдам жұмыс істеуіне көптеген мүмкіндіктер береді. Параллель есептеулерді қарастыра отырып, қазіргі таңда параллель есептеулерсіз төменде аталғандай интерактивті сахналардың қолданушымен өзара байланысын жүзеге асыру мүмкін емес:

- Қолданушымен енгізілген арнайы форматталған мәтіндер. Ондай форматтаудың мысалы- бағдарлама кодында синтаксистік қателерді ерекшелеу немесе мәтінде орфографиялық және пунктуацияық қателеді ерекшелеу. Мұндай ерекшелеулер параллель ағындарда қолданылады;
- Қолданушының күрделі сұраныстарын орындау барысында олардың аяқталуы туралы ескерте отырып орындау. Интернеттен файлдарды көшіру дәл аталған қағида бойынша жүзеге асырылады.
- Геоақпараттық жүйелерде жүктелу мен кеңістікті мәліметтерді бейнелеу көптеген уақытты алады. Көбінесе көлемді қабаттарды параллельді үрдіс ерекшелейді, ал қолданушы жүйемен жұмыс істеу мүмкіндігіне бірден ие болады.

Қорытындылай келе, параллель есептеулер қазіргі таңда жоғарыда аталғандай көптеген мүмкіндіктерге ие және де кез келген ортада қолданылады. Ал, осы паралель есептеулерді жүзеге асырудың бастамасында жоғарыда аталған әдістерді қолданбай іске қосу мүмкін емес.

Қолданылған әдебиеттер тізімі

1. Словарь по кибернетике / Под редакцией академика В. С. Михалевича- 2-е.-Киев, 1989. -751 с.
2. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — СПб: БХВ-Петербург, 2002. — 608 с.
3. Тихонов, И. В. Параллельное программирование в .NET. Иркутск, 2009.