ПРОГРАМНО-ТЕХНІЧНІ КОМПЛЕКСИ

А.В. АНИСИМОВ, А.М. ФЕДОРУС

УДК 004.75

РАЗРАБОТКА И ПЕРСПЕКТИВЫ СИСТЕМЫ ПАРУС-WCF

Аннотация. Рассмотрены детали реализации системы ПАРУС-WCF. Система ПАРУС-WCF обеспечивает создание и функционирование асинхронно-рекурсивных процессов при решении задач параллельных вычислений как на одном ПК, так и на компьютерной сети. Она позволяет разным модулям обмениваться информацией, обеспечивая обмен данными в системах с распределенной и нераспределенной памятью.

Ключевые слова: ПАРУС-WCF, ПАРУС, параллельные вычисления, система распределенных вычислений.

ВВЕДЕНИЕ

Цель данной статьи — изучение и реализация системы вычисления параллельных алгоритмов на базе концепции ПАРУС (Параллельные Асинхронные Рекурсивно Управляемые Системы). Реализация такой системы целесообразна, поскольку может быть применена для широкого круга задач. Примерами таких задач могут быть моделирование физических процессов, обработка графики, решение уравнений, множество научных, инженерных задач, а также задач бизнеса.

В настоящее время имеется много программных комплексов для реализации параллельных алгоритмов (ПА) и создания параллельных систем (ПС). Однако существует недостаточно адекватных средств описания ПА с динамической структурой для запуска в ПС. Язык C# (который будет использован для реализации ПАРУС-WCF) имеет много встроенных средств реализации параллелизма, но ни один из них не дает возможности перенести эти вычисления на компьютерную сеть.

Система программирования ПАРУС — это множество программных средств, которые обеспечивают процесс разработки, реализации и функционирования алгоритмов параллельной обработки информации вследствие расширения некоторого базового языка. Основное преимущество использования концепции ПАРУС — это заложенные в нее концепции рекурсии, асинхронности и бесконечного параллелизма. Благодаря этому любой параллельный алгоритм легко масштабируется на любом множестве вычислительных средств.

Реализация системы ПАРУС-WCF ориентирована на создание простого инструмента для организации распределенных (в первую очередь) вычислений; поэтому основная часть работы посвящена обзору технологии WCF, ее особенностям и организации сетевого взаимодействия. При разработке уделено внимание созданию простых интерфейсов для конечного пользователя.

ПАРУС-ТЕХНОЛОГИЯ

ПАРУС-технология программирования представляет некоторое множество программных средств, обеспечивающее процесс разработки и реализации алгоритмов параллельной обработки информации [1, 2], и базируется на концепции

управляющего пространства (УП), предложенного в [3], и которое представляется виртуально параллельным пространством (ВПП) [4]. УП можно представить динамическим графом, который используется для описания логической и коммуникационной структуры задачи и отображения динамических изменений в ней. Благодаря базовым концепциям ПАРУС-приложений система позволяет:

— описывать процессы со статическим и динамическим параллелизмом;

— использовать рекурсию по данным и по управлению;

— в явном виде описывать распределение ресурсов, схемы связи между элементами, логическую структуру системы;

— строить систему виртуальных ПАРУС-машин, что позволяет моделировать распределенные процессы на любом количестве вычислительных машин.

Основные термины ПАРУС-технологии: точка, программный канал (ПК), алгоритмический модуль (АМ).

Управляющее пространство представляет граф, вершины которого — точки, а ребра представляют программные каналы. При этом две точки могут соединяться разными каналами. Каждая точка представляет собой поток, который выполняет алгоритмический модуль. Она может самостоятельно создавать новые точки и изменять УП. АМ выполняются параллельно и могут обмениваться данными через связи между точками.

ВЫБОР ТЕХНОЛОГИИ ПЕРЕДАЧИ ДАННЫХ

Выбор способа передачи данных между точками – один из главных критериев оценки работы ПАРУС. Для создания разнообразных API для обмена данными платформа .Net Framework предоставляет одновременно несколько разных технологий:

— Windows Communication Foundation (WCF),

— ASP.NET WebAPI,

— ASP.NET Core MVC.

Для выбора оптимального варианта нами проведено серию тестов. Существуют два критерия оценки: среднее время операции и количество использованной оперативной памяти. Все три технологии применяют протокол HTTP, как единственно общий. В каждом случае использовались стандартные настройки формата передачи (текстовые SOAP XML для WCF и JSON для остальных технологий). Функционал каждого сервиса был чрезвычайно прост: получить массив значений и его размер, а затем отправить его обратно.

Для тестов используем два типа данных: малый объект, представленный одним полем типа Guid, и большой объект, который имеет десять полей разных типов; три из них содержат другие объекты. Также мы изменяли количество таких объектов в массиве. Все тесты проводились на одном и том же ПК с процессором AMD A10-6790K и версией .NET Framework 4.7 (CLR 4.0.30319.42000), 64bit RyuJIT-v4.7.2650.0.

В табл. 1 приведены результаты обмена с выбранными серверами с пустым массивом. Как видим, сеть WCF BasicHttpBinding заметно быстрее, чем методы ASP.NET WebAPI Json и ASP.NET Core Json, хотя все они работают с одинаковым протоколом. Объем используемой памяти также ниже. Аналогичная ситуация, но при числе объектов, равным 100, прослеживается и в табл. 2. Метод WCF NetTcpBinding был использован для дополнительной демонстрации превосходства WCF, которое получаем с помощью более простого протокола. Метод WCF NetTcpBinding не использует протокол HTTP над TCP и благодаря этому выигрывает и по скорости, и по памяти у аналогов, использующих HTTP. При этом для разработчика нет разницы, какой протокол задействован.

Технология WCF была специально разработана для построения производительных API и распределенных систем. В отличие от остальных интерфейсов, которые использовались в прошлом (например, DCOM, .NET Remoting и другие

Таблица 1

Метод	Результаты с пустым массивом малых / больших объектов	
	Среднее время запроса (мкс)	Объем памяти (Кб)
WCF BasicHttpBinding	368.4 / 385.6	28.1 / 28.12
WCF NetTcpBinding	103.6 / 107.8	4.64 / 4.64
ASP.NET WebAPI Json	499.0 / 545.4	69.67 / 69.55
ASP.NET Core Json	534.8 / 570.7	44.26 / 44.26

Таблица 2

Метод	Результаты с массивом малых / больших объектов	
	Среднее время запроса (мкс)	Объем памяти (Кб)
WCF BasicHttpBinding	1018.6 / 12705.6	89.81 / 1264.54
WCF NetTcpBinding	617.4 / 6796.2	59.54 / 708.08
ASP.NET WebAPI Json	1674.7 / 19177.9	208.99 / 2657.31
ASP.NET Core Json	1531.2 / 24834.6	163.13 / 2354.72

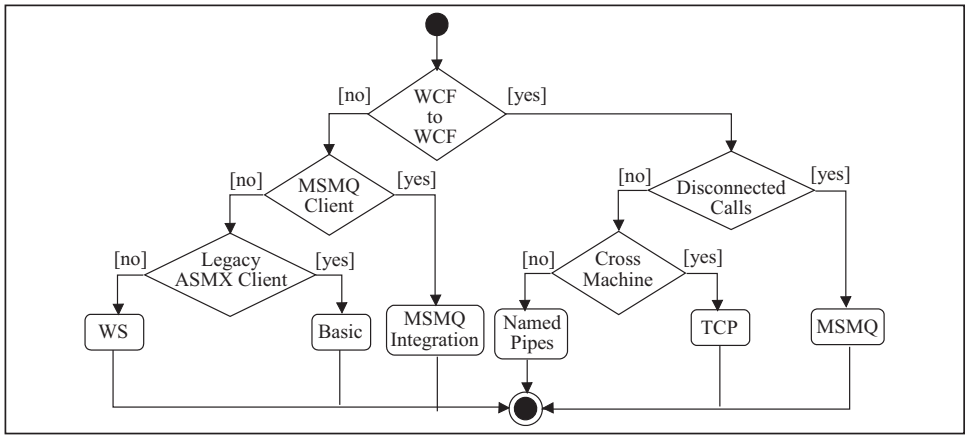

Рис. 1. Схема выбора привязки WCF

веб-службы XML), WCF предлагает единый, унифицированный программный интерфейс, который может использоваться для взаимодействия с множеством разнообразных сетевых технологий. Другими словами, одна и та же реализация программного интерфейса может использоваться для обработки абсолютно разных протоколов. Изложим основные понятия в WCF.

— **Адрес:** описывает расположение службы.

— **Привязка:** в WCF имеется порядка десяти разнообразных протоколов передачи данных, которые определяются сетевым протоколом, механизмом кодирования и транспортным уровнем.

— **Контракт:** описывает каждый метод, который предоставляется службой WCF.

Существует множество разнообразных привязок, общий алгоритм выбора которых представлен схемой на рис. 1.

ПАРУС-WCF

При разработке системы сделан акцент на создание простого интерфейса для разработчика и на оптимизацию сетевых взаимодействий. При этом методы библиотеки классов — асинхронные, что дает более гибкие возможности

в контроле выполнения операций, например рассылка данных по точкам, создание новых точек или другие взаимодействия с сетью.

Для обмена данными между точками используется WCF-веб-сервис с привязками netTcpBinding и netNamedPipeBinding. Каждый метод взаимодействия используется для оптимизации разных случаев передачи данных. Например, привязка netNamedPipeBinding использует технологию именованных каналов операционной системы Windows, которая обеспечивает обмен данными между разными процессами внутри одной машины. Передача данных происходит за счет общей памяти ядра операционной системы и является более быстрым вариантом, чем использование протокола TCP/IP. Кроме обеспечения взаимодействия между точками эта привязка используется для взаимодействия с демонами.

Разработанная система включает демон и библиотеку классов.

Daemon (или демон) — клиент, установленный на каждую машину, которая будет использована в вычислениях. Демон по сути является WCF-веб-сервисом, который может принимать файлы и запускать АМ. Одновременно с программой (или службой) демона запускаются два веб-сервиса WCF, которые обеспечивают взаимодействия между точками. Один из этих сервисов реализует программные каналы между точками.

Библиотека классов содержит необходимые абстракции, которые используются при написании кода для АМ.

Начало любого ПАРУС-процесса — это создание УП и указание адресов машин, которые будут использованы в вычислениях:

```
var cs = new ControlSpace("TaskName", new List<string>( ) {"ip:port", ...)};
```

Объект управляющего пространства кроме объединения машин в группу выполняет несколько функций. Во-первых, при уничтожении этого объекта использованные демоны получат уведомление, что управляющее пространство прекращает свою работу и необходимо уничтожить остальные его точки. Во-вторых, поскольку создание точек (или подключение к существующим) происходит через объект УП, нам не нужно явно указывать, какая точка отправила данные. Именно эти проблемы решает объект УП. Используя этот объект, пользователь создает точки, задает точкам их уникальные идентификаторы и идентификаторы точек, с которыми может взаимодействовать:

```
var point = await cs.CreatePointAsync();

var point = await cs.CreatePointAsync(Name, PointType.Any, ChannelType.TCP);
```

При создании точки пользователь может указать дополнительные параметры.

Name — название точки, позволяющее искать ее в списке каналов.

PointType — тип расположения точки; параметр принимает такие значения:

— PointType.Any — значение по умолчанию: в таком случае объект УП выбирает тип расположения точки самостоятельно.

— PointType.Local — созданная точка будет расположена в локальном демоне; оптимальный вариант, если АМ будет выполнен быстро.

— PointType.Remote — точка будет создана на удаленной машине (по возможности).

Третий параметр (ChannelType) позволяет явно указать приоритетный тип канала к этой точке. При передаче данных система может самостоятельно выбирать оптимальный вариант передачи, но при необходимости симулировать работу компьютерной сети на одной машине. Для этого необходимо явно указать тип канала TCP, который принимает три значения: ChannelType.Any, ChannelType.TCP, ChannelType.NamedPipe.

Кроме того, УП позволяет распространять файлы АМ и получать информацию о демонах:

```
bool done = await cs.AddFileAsync(new FileInfo(“pathToFile”));
done = await cs.AddDirectoryAsync(new DirectoryInfo(“pathToDirectory”));
IEnumerable<Daemon> daemons = cs.Daemons;
```

После создания точки возможны следующие взаимодействия с ней:

— добавить канал с существующей точкой;

— передать данные;

— запустить в этой точке некоторый АМ;

— ожидать от нее данные;

— остановить выполнение АМ.

Запуск точки происходит с помощью метода

```
await point.RunAsync(new PointStartInfo(Method));
```

В конструктор класса PointStartInfo передается метод. Данные о методе и сборке, в которой он находится, будут автоматически собраны с помощью рефлексии. В случае нестатического класса, в котором находится этот метод, будет вызван пустой конструктор. Метод должен соответствовать следующему критерию: принимать первым параметром экземпляр класса PointInfo и возвращать Task. Экземпляр класса PointInfo создается демоном при запуске АМ. В нем хранятся данные о точке, запустившей этот АМ, определенная информация об УП, которому она принадлежит, и список каналов, переданных в эту точку.

Передача данных между точками происходит с помощью следующих методов:

```
Task SendAsync<T>(T data);
Task<T> GetAsync<T>( );
```

Здесь T — некоторый тип. Обобщенные типы (Generics) — одна из особенностей языка C#, благодаря которой можно создавать классы и методы, использующие неизвестные при компиляции типы. Классы, которые будут переданы в данный метод перед передачей в другую точку, будут сериализованы библиотекой Json.NET. Этот процесс преобразует открытые поля объекта в данные формата Json и они позже будут переданы в другую точку. Использование внешнего механизма сериализации обусловлено следующим.

— Встроенный сериализатор требует атрибутов [Serializable] или [DataContract] для маркировки данных. Это ограничивает использование существующего кода и готовых библиотек.

— Веб-сервисы WCF требуют четко сформированных интерфейсов данных. Единственный способ передачи данных, о которых сервис не знает, — конвертация в один из известных ему типов. В нашем случае данные пользователя представляются как массив байтов.

— Для других технологий передачи данных неоднократно проводились тесты производительности и выбор протокола, который оказывал сильное влияние на результат.

— Передача данных между точками происходит асинхронно, и после отправки сохраняются в оперативной памяти точки-получателя.

Получение данных точкой имеет следующую оптимизацию. Поскольку процесс веб-сервиса и поток выполнения привязанной к нему точки имеют общую память (в виде экземпляра класса PointInfo), WCF-сервис использует механизм событий (event) и уведомляет о получении новых данных экземпляр класса Point. Таким образом, в точку приходят данные. Если АМ не ожидает получение этих данных, то они будут сохранены в классе PointInfo. В противном случае данные будут сразу получены и выполнение АМ продолжится. Такой механизм намного лучше, чем периодический опрос сервиса.

СРАВНЕНИЕ СИСТЕМЫ ПАРУС-WCF С ДРУГИМИ РЕАЛИЗАЦИЯМИ ПАРУС

В настоящее время уже существуют несколько реализаций ПАРУС. Каждая система имеет свой подход к реализации концепций ПАРУС. Первые реализации ПАРУС — PARCS-PASCAL [1], PARCS-Modula2 [5], PARCS-FORTRAN [6], PARCS-C [6] были заточены под использование одной машины. Затем перешли на использование компьютерных сетей — PARCS-Java [7–9], PARCS-Cuda [10], а системы PARCS-Python [11], PARCS-C# [12] ориентированы на использование в облачном окружении. ПАРУС-WCF может применяться в любом из этих сценариев.

Система ПАРУС-Java является наиболее популярной и распространенной реализацией ПАРУС на данный момент. ПАРУС-WCF в сравнении с ПАРУС-Java имеет следующие особенности.

— Наличие асинхронных интерфейсов.

— Оптимальное использование сетевых ресурсов. В ПАРУС-Java каждая точка привязана к некоторому сокету, и обмен данными происходит при установке контакта. При таком подходе одна из точек постоянно пытается подключиться к другой или ожидает информацию. ПАРУС-WCF использует два сокета: один для демона, а другой для сервиса-точек. Передача данных между точками происходит через сервис-точек и данные всегда передаются. Таким образом, блокировка происходит только при ожидании данных.

— Оптимизация передачи данных. При возможности передачи данных внутри машины ПАРУС-WCF не выходит на уровень TCP-IP. Пользователь может самостоятельно выбирать средства сериализации данных.

— Существует возможность рекурсивно создавать точки.

ПРИМЕНЕНИЯ ПАРУС-WCF

Концепция ПАРУС позволяет реализовывать алгоритмы, которые используют все виды декомпозиции (данных, функций и их комбинаций). Наиболее распространенные примеры — алгоритмы вида «разделяй и властвуй», например классический частично-рекурсивный параллельный алгоритм умножения матриц, который демонстрировался в [7], или полностью рекурсивная версия в [13]. Такие алгоритмы полностью раскрывают концепции рекурсии и неограниченного параллелизма.

Алгоритмы такого типа в общем случае отражены в рис. 2. Прямоугольники отражают точки, а стрелки — каналы. Главный модуль начинает рекурсивное разбиение задачи и запускает алгоритмические модули, каждый из которых может, в свою очередь, стать главным и разделить свою задачу на части. Эта схема представляет собой управляющее пространство этой задачи, а каждый алгоритмический модуль может рекурсивно создавать подпространство для выполнения своих задач.

Примером реализации такого алгоритма может быть представленная в работе [14] рекурсивная реализация перемножения матриц. Используя классическую формулу блочного вычисления произведения матриц, получаем

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}. \quad (1)$$

С помощью формулы (1) мы рекурсивно сокращали размерность производимых матриц до выбранного минимального размера и при возврате рекурсии выполняли суммирование соответствующих элементов.

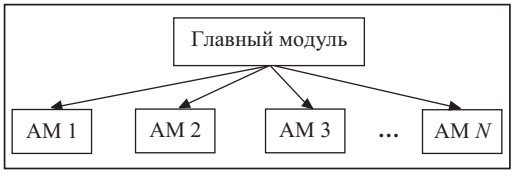

Рис. 2

Рис. 3 демонстрирует УП при раскрутке рекурсии, данные о матрицах из начального модуля переда-

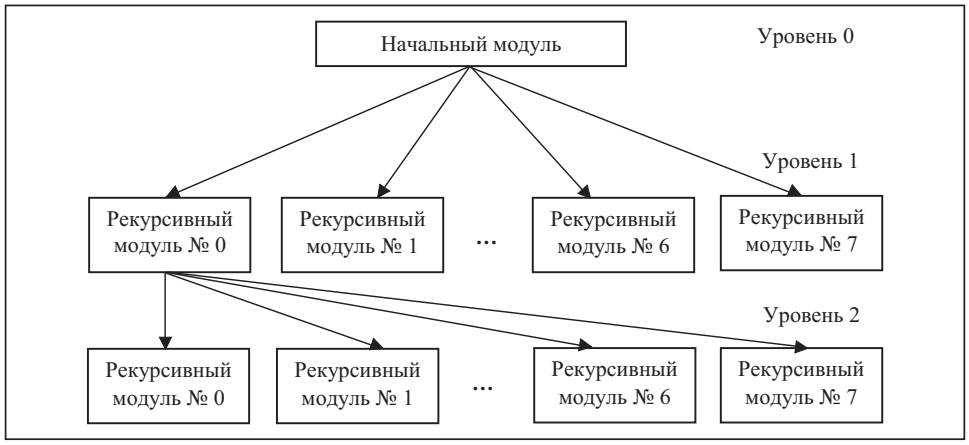

Рис. 3

ются по каналам вниз, рекурсивные модули при этом, если матрица больше минимального размера, ведут себя как начальный модуль и передают вычисления в «свои» новые рекурсивные модули.

Далее наступает этап сбора результатов, и тогда может быть использовано два подхода к передаче матриц вверх: передача полных матриц и передача фрагментов матриц минимального размера. Для случая с матрицами минимального размера получаем следующий код:

```
for (int j = 0; j <getMatrixNumber(size); j++) {
    for (int i = 0; i < 8; i += 2) {
    var firstPart = await points[i].GetAsync<MatrixPart>( );
    var secondpart = await points[i+1].GetAsync<MatrixPart>( );
    firstPart.matr.Add(secondpart.matr);
    info.ParentPoint.SendAsync(firstPart);
}}
```

В начальном модуле в последней строке кода вместо отправки будет запись в новый объект матрицы.

На матрицах размера $2^N \times 2^N$ разрядность можно уменьшать вплоть до матриц 2×2, однако в какой-то момент затраты на передачу данных и создание новых точек превышают затраты на непосредственное вычисление произведения матриц.

ЗАКЛЮЧЕНИЕ

В статье приведены способ и технические детали реализации системы ПАРУС-WCF языком C#. Данная реализация может быть использована разными языками платформы .Net и привязана к технологии WCF, которая не является мультиплатформенной, и возможности ее переноса на другие платформы (вместе с развитием .Net Core) пока не планируется. Но поскольку демоны являются WCF-сервисами, их можно динамически создавать в облачном сервисе Azure.

Особо следует отметить результаты тестирования разных технологий веб-сервисов. Они показали эффективность использования WCF. Также отметим большое количество настроек для привязок WCF и возможность простой сменой настроек обеспечить безопасность передачи данных (особенно если вычисления происходят в публичной сети или облаке) и их сжатие.

СПИСОК ЛИТЕРАТУРЫ

1. Анисимов А.В., Кулябко П.П. Программирование паралельных процессов в управляющих пространствах. *Кибернетика*. 1984. № 3. С. 79–88.

2. Анисимов А.В., Кулябко П.П. Особенности ПАРУС-технологии. *Кибернетика и системный анализ.* 1993. № 3. С. 128–137.
3. Глушков В.М., Анисимов А.В. Управляющие пространства в асинхронных параллельных вычислениях. *Кибернетика*. 1980. № 5. С. 1–9.
4. Анісімов А.В., Ветров А.М., Кулябко П.П. Керуючий простір як віртуально паралельний простір. *Вісник Київського університету. Сер. фізико-математичні науки.* 1999. № 4. С. 82–86.
5. Анисимов А.В., Кулябко П.П. Моделирование сети Петри с помощью ПАРУС-средств. *Проблемы программирования.* 1997. Вып. 2. С. 45–56.
6. Анисимов А.В., Борейша Ю.Е., Кулябко П.П. Система программирования ПАРУС. *Программирование*. 1991. № 6. С. 91–102.
7. Анісімов А.В., Дерев'янченко О.В. Система ПАРКС-JAVA як засіб вирішення паралельних алгоритмів на комп'ютерній мережі. *Проблеми програмування*. 2004. № 2, 3. С. 282–284.
8. Дерев'янченко О.В. Моделювання паралельних програм за допомогою системи ПАРКС-JAVA *Наукові записки НаУКМА. Комп'ютерні науки*. 2005. Т. 36. С. 32–38.
9. Anisimov A.V., Derevianchenko A.V., Kuliabko P.P., Fedorus O.M. Programming system PARCS. *Journal of Computer and Communications*. 2017. Vol. 5, N 9. P. 129–139.
10. Деревянченко О.В. Застосування технології CUDA в системі паралельних обчислень ПАРКС-Java. *Матеріали міжнародної конференції «Штучний інтелект. Інтелектуальні системи ШІ-2011».* Донецьк: ІПШІ «Наука і освіта». 2011. Т. 1. С. 62–68.
11. Анісімов А.В., Кулябко П.П., Годованюк М.І. Паралельне програмування в мережах на основі ПАРКС-технології (базова мова Python). *Вісник Київського національного університету імені Тараса Шевченка. Сер. фізико-математичні науки*. 2016. № 3. С. 57–60.
12. Деревянченко О.В., Хавро А.Ю. Застосування системи ПАРКС.NET та Amazon EC2 для хмарних обчислень. *Вісник Київського національного університету імені Тараса Шевченка. Сер. фізико-математичні науки.* 2015. № 4. С. 111–118.
13. Федорус О.М. Системи ПАРКС як засіб реалізації хмарних обчислень. *Інформаційні технології та взаємодії (IT&I-2015): Тези доп. II Міжнар. наук.-практ. конф*. Київ, 3–5 листопада 2015 р. Київ: Вид.-поліграф. центр «Київ. ун-т», 2015. С. 115–116.
14. Федорус О.М. Застосування ПАРКС-C# для моделювання паралельно-рекурсивних процесів. *Вісник Київського національного університету імені Тараса Шевченка. Сер. фізико-математичні науки.* 2017. № 1. С. 75–79.

Надійшла до редакції 14.08.2019

А.В. Анісімов, О.М. Федорус
РОЗРОБКА ТА ПЕРСПЕКТИВИ СИСТЕМИ ПАРКС-WCF

Анотація. Наведено деталі реалізації системи ПАРКС-WCF. Система ПАРКС-WCF забезпечує створення та функціонування асинхронно-рекурсивних процесів для розв'язання задач паралельних обчислень як на одному ПК, так і на комп'ютерній мережі. Вона дозволяє різним модулям обмінюватися інформацією, забезпечуючи обмін даними в системах з розподіленою і нерозподіленою пам'яттю.

Ключові слова. ПАРКС-WCF, ПАРКС, паралельні обчислення, система розподілених обчислень.

A.V. Anisimov, O.M. Fedorus
DEVELOPMENT AND PROSPECTS OF THE PARCS-WCF SYSTEM

Abstract. The paper presents details of implementation of PARCS-WCF system. The PARCS-WCF system provides creation and operation of asynchronous-recursive processes in solving parallel computing tasks both on a single PC and on a computer network. It allows different modules to exchange information providing data exchange in systems with distributed and non-distributed memory.

Keywords: PARCS-WCF, PARCS, parallel computing, distributed computing system.

Анисимов Анатолий Васильевич,
доктор физ.-мат. наук, чл.-кор. НАН Украины, профессор кафедры Киевского национального университета имени Тараса Шевченко, e-mail: avatatan@gmail.com.

Федорус Алексей Мстиславович,
магистр, ассистент кафедры Киевского национального университета имени Тараса Шевченко, e-mail: alex.fedorus@gmail.com.