ОРГАНИЗАЦИЯ СИСТЕМЫ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ ВИБРОМЕТРИЧЕСКИХ ДАННЫХ

В.Э.Базаревский

Белорусский государственный университет информатики и радиоэлектроники,
Кафедра программного обеспечения информационных технологий
ул. Гикало 9, Минск, Республика Беларусь
телефон: + (37529) 5122209; e-mail: baz-val@yandex.ru
web: www.vibrosignal.com

Рассматриваются тенденции развития современных измерительно-вычислительных комплексов, возможность использования специализированных сред для параллельной обработки сигнальных данных в таких комплексах, средства и способы их интеграции с программными средствами общего назначения.

Распределенные системы, параллелизм, виброметрия, координационные языки

1 ВВЕДЕНИЕ

Широкое распространение мобильных компьютеров различного исполнения, и возможность комплектования их типизированными модулями аналого-цифрового преобразования позволяет создавать малогабаритные системы, ориентированные на решение задач оценки физических параметров различных объектов. Однако следует заметить, что определенный класс задач по обработке данных не может быть решен в приемлемый промежуток времени применением таких систем ввиду их довольно ограниченной вычислительной мощности. Для решения этих задач предлагается создание централизованного вычислительного кластера. При возникновении подобных задач, данные с таких мобильных систем, могут быть посланы на кластер, обработаны и пересланы обратно. Ввиду неполной занятности кластера обработкой запросов от одной из таких мобильных систем на протяжении всей его работы, использование кластера представляется эффективным.

2 ВЫБОР АППАРАТНОГО ОБЕСПЕЧЕНИЯ

Создание выделенного кластера требует наличия значительного числа электронно-вычислительных машин, имеющую сходную аппаратную архитектуру, с совместимыми операционными системами, соединенных в одну локальную сеть, что, в конечном итоге, требует выделения отдельного помещения, средств охлаждения серверов, обслуживающего персонала. Следует заметить, что большинство организаций, имеют достаточно большой парк персональных компьютеров, предназначенных для работы на них офисными работниками, при этом, обычно, процент загруженности этих компьютеров составляет 5-10% максимальной мощности, что делает возможным использование этих компьютеров в качестве узлов кластера, путем добавления низкоприоритетного процесса, выполняющего необходимые вычисления. Таким образом, создание подобного кластера на базе существующего парка офисных компьютеров делает его условно бесплатным.

3 ВЫБОР ВЫЧИСЛИТЕЛЬНОЙ СРЕДЫ

Так как, основные вычислительные мощности кластера базируются на парке офисных компьютеров, не стандартизованных по типу аппаратного и программного обеспечения, вычислительная среда подобного кластера должна быть кроссплатформенной. Помимо кросплатформенности, такая вычислительная среда должна поддерживать распределенные вычисления, создание нового процесса как на той же машине, так и на другой должно иметь схожий синтаксис, и по возможности автоматизировано.

На основе анализа ряда зарубежных источников [1] [2] [3] и тенденций развития параллельного программирования [4], в связи с проблемами, возникающими при синхронизации потоков, возникновением взаимных блокировок, ресурсоемкостью создания новых потоков и сложностью распараллеливания императивных программ, сделан вывод о нецелесообразности использования модели потоков, и использующих эту модель сред общего назначения, таких как .Net Framework и Java.

Среди существующих вычислительных сред, поддерживающих распределенные вычисления, имеющих расширенный функционал по созданию новых процессов и обмену сообщениями между ними, можно выделить среду и язык Erlang, поддерживающий модель легковесных процессов, широко применяемый в сфере телекоммуникаций и имеющий эффективную реализацию обмена сообщениями между процессами как на локальном компьютере, так и на другом вычислительном узле. Кроме того, среда и язык Erlang свободны для распространения и имеют реализации для большинства современных операционных систем, таких как Unix, Windows, MacOS.

Следует заметить, что для создания удобных для понимания, поддержки и последующего расширения распределенных программных средств необходимо расширение языка Erlang для неявного распараллеливания алгоритмов. Подобным функционалом обладает язык программирования Lisp2D, но в отличие от Erlang, он не получил широкого распространения, имеет слабую интеграцию со средами общего назначения, такими как .Net и Java, а соответственно его использование не представля-

ется эффективным из-за необходимости создания большого числа дополнительного функционала.

Суть расширения заключается в том, что по умолчанию, при создании нового процесса, для вычисления нового процесса необходимо передать имя функции для исполнения и аргументы, а по окончании выполнения переданной функции процесс должен переслать результаты выполнения этой функции, а вызывающий процесс должен получить эти результаты. Причем, вышеописанный алгоритм вычисления функции в отдельном процессе должен быть выполнен для каждой функции, тем самым, значительно усложняя структуру программы и усложняя ей поддержку и изменение. Кроме того, подобная схема требует внесения функционала, связанного с распараллеливанием, в тело функции, где он является инородным. Подобный способ распараллеливания не представляется оптимальным, так как значительно усложняет программу, повышает вероятность возникновения новых ошибок в алгоритмах и делает невозможным повторное использование вызываемых функций.

Для решения вышеописанных проблем предлагается создание отдельного модуля, инкапсулирующего операции создания новых процессов, передачи и получения сообщений. Таким образом, при необходимости параллельного выполнения функций, в специализированную функцию передается словарь, содержащий ключи, по которым можно по окончании вызова получить результаты выполнения функций, функции и аргументы. Диаграмма последовательности вызова функций параллельно показана на рисунке 1.

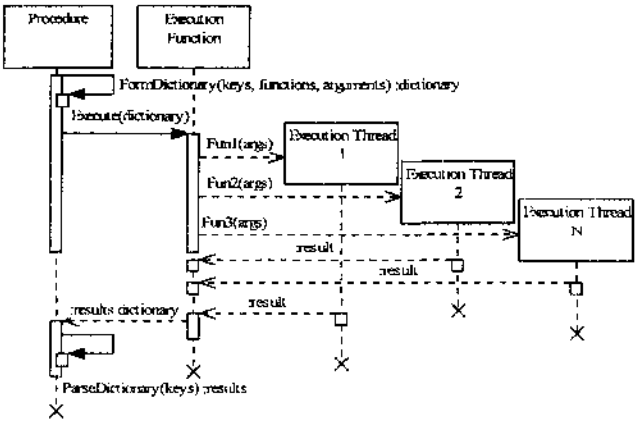

Рис.1. Диаграмма последовательности вызова функций при неявном использовании распараллеливания.

Таким образом, при необходимости вычисления некоторого числа функций в параллельном режиме, разработчику необходимо сформировать словарь, содержащий ключи, идентификаторы функций и аргументы, необходимые для вычисления этих функций (На диаграмме эти действия производятся в функции FormDictionary). После, когда такой словарь сформирован, необходимо вызвать функцию Parallel:Execute и передать в неё в качестве аргумента сформированный словарь, который обрабатывается, и для каждого элемента словаря создается процесс, который выполняет соответствующую ему функцию, и, по окончании выполнения функции, посылает сообщение родительскому процессу, о том, что функция выполнена и передаёт результаты выполнения этой функции, родительский процесс обрабатывает полученный результат, и добавляет в результирующий словарь пару ключ – результат выполнения. По получении сообщений о выполнении от всех дочерних процессов, функция Parallel:Execute завершает свою работу, и возвращает сформированный результирующий словарь.

Помимо указанных действий, подобный функционал может быть расширен, добавлением алгоритма определения целесообразности выполнения той или иной функции в процессе на другом компьютере.

Таким образом, создание программного обеспечения с использованием специализированных сред, направленных на эффективное распараллеливание алгоритмов представляется целесообразным. А использование в качестве базовой вычислительной мощности парк офисных компьютеров в качестве вычислительно кластера обуславливает экономическую эффективность подобного решения. Кроме того, функционал для неявного распараллеливания может быть использован при создании реализаций координационных языков в платформе Erlang, таких как ErLinda [5].

ЛИТЕРАТУРА

[1] Edward A. Lee. The Problem With Threads // University of California at Berkeley. Technical Report No. UCB/EECS-2006-1 – 10 Jan 2006. http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-1.html

[2] Edward A. Lee. Disciplined Message Passing // University of California at Berkeley. Technical Report No. UCB/EECS-2009-7 – 18 Jan 2009. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-7.html

[3] Хоар Ч.Э. Взаимодействующие последовательные процессы: Пер. с англ. – М: Мир, 1989 г. -264 с.

[4] Yang Zhao. On the Design of Concurrent, Distributed Real-Time Systems // University of California at Berkeley Technical Report No. UCB/EECS-2009-117 http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-117.html 13 Aug 2009

[5] G. C. Wells. Linda implementations in Java for concurrent systems // CONCURRENCY | PRACTICE AND EXPERIENCE Concurrency: Pract. Exper. 2003;