Сравнение использования технологий параллельного программирования Microsoft применительно к задаче поиска данных MapReduce

Г. Г. Федюкович

Сформулирована задача поиска данных MapReduce. Разработаны реализации на языках C++ и C# и F# с использованием технологий OpenMP, Threading, TPL, PPL. Приведены результаты исполнения для различных объемов входных данных.

В статьях "Future of Computer Architecture"Дэвида Паттерсона [1] и "A View From Berkeley" ученых Университета Беркли [2] задача MapReduce [3] была отнесена к списку широко известных и применяемых алгоритмов, использующих "шаблонные" вычисления и коммуникацию между реализующими их процессами. Цели данного исследования - разработка и реализация параллельных алгоритмов решения рассматриваемой задачи на различных языках программирования и с использованием различных технологий параллельного программирования, сравнение полученных результатов по критериям быстродействия, и удобства программирования.

Основная часть работы была выполнена в рамках проекта "Parallel Dwarfs" (http://paralleldwarfs.codeplex.com/) под руководством Microsoft HPC [4].

Рассмотрим решение задачи подсчета числа вхождений слов в данный текст. Множество входящих данных – упорядоченное индексированное множество символов. Функция Map находит среди текста слова – последовательности символов, состоящие только из букв, - и возвращает множество слов в виде списка. Далее функция Reduce сортирует список и создает список пар (слово, количество его вхождений) за один его обход.

В работе показано, что использование технологий параллельного программирования улучшает производительность алгоритмов решения данной задачи. Алгоритмы разработаны на языках C++, C# и F#. В каждую из реализаций внедрены элементы распараллеливания решения подзадач разбиения входных данных, распределения вычислений на процессоры, объединения результатов. Для C++ использовалась классическая технология OpenMP и технология Parallel Patterns Library (PPL), для C# - традиционные методы создания потоков (Threading) и пула потоков (ThreadPool), и более новая технология Task Parallel Library (TPL), для F# - TPL.

Сборка приложений осуществлялась средствами MS Visual Studio 2010 Beta 2. Тестирование приложений происходило на рабочей станции: Laptop, 3GB RAM, Intel Core2 Duo 2,26 HGz CPU, Vista SP2, MS .Net Framework 4 Extended Beta 2 и MS Visual C++ 2010 Beta 2 x86 Runtime.

Данные	C++			C#				F#	
	Serial	PPL	OMP	Serial	TPL	Thread	TPool	Serial	TPL
25 Mb	10.353	6.324	6.145	12.847	8.858	7.900	7.987	14.472	8.560
50 Mb	23.232	13.825	13.735	27.601	18.024	16.316	16.761	32.193	18.364
100 Mb	50.877	32.083	32.645	59.529	37.915	37.520	38.217	68.251	39.445

Таблица 1. Среднее время исполнения (сек) для C++, C# и F#

За пределами рассмотрения пока остались многие другие средства, такие как интерфейс пересылки сообщений MPI, которые заслуживают отдельных статей. Опыт и позитивные результаты дают нам больше возможностей и перспективных тем для дальнейших исследований.

Литература

1. D. A. Patterson. "Future of Computer Architecture" 2006.
2. K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. Webb, Williams, K. A. Yelick. "The Landscape of Parallel Computing Research: A View From Berkeley". University of California, 2006
3. J. Dean, S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters" Google, 2004
4. S. Mortazavi, J. Baxter, "Building Supercomputer Applications using Windows HPC 2008", 2008