УДК 519.7, 519.17

ПРОГРАММНАЯ СРЕДА ДЛЯ РАБОТЫ С КЛАСТЕРНЫМ КОМПЬЮТЕРОМ ИЗ ОС WINDOWS

Д.И. Черемисинов

ведущий научный сотрудник ОИПИ НАНБ
кандидат технических наук, доцент

Л.Д. Черемисинова

главный научный сотрудник ОИПИ НАНБ
доктор технических наук, профессор

Объединений институт проблем информатики Национальной академии наук Беларуси, Республика Беларусь
E-mail: {cher, cld}@newman.bas-net.by

Д.И. Черемисинов

Окончил Томский *государственный* университет, *кандидат технических наук, доцент. Работает в ОИПИ НАН Беларуси в должности ведущего научного сотрудника и Белорусском государственном университете информатики и радиоэлектроники в должности доцента.*
Круг научных интересов: программирование, логическое проектирование и тестирование дискретных систем управления, реализация параллельных алгоритмов управления.

Л.Д. Черемисинова

Окончила Томский *государственный* университет, *доктор технических наук, профессор. Работает в ОИПИ НАН Беларуси в должности главного научного сотрудника и Белорусском государственном университете информатики и радиоэлектроники в должности профессора. Круг научных интересов: дискретная математика, логическое проектирование и тестирование дискретных систем управления, реализация параллельных алгоритмов управления*.

Аннотация. Рассматривается проблема разработки и выполнения программ для мультипроцессорных систем кластерного типа. При разработке программ для кластерного компьютера применяется технология, основанная на использовании удаленного терминала. Рассматривается ситуация, когда таким удаленным терминалом является компьютер с операционной системой Windows. Предлагается набор инструментальных средств, позволяющий выполнять задачи редактирования текста, компиляции программы и запуска программы на кластере. Достоинством предлагаемого способа подготовки программы к выполнению является то, что она позволяет программисту, имеющему опыт работы с инструментами для Windows, использовать эти знания при разработке программ для кластерного компьютера, работающего в среде UNIX.

Ключевые слова: параллельные вычисления, кластерный компьютер, технология программирования.

Введение.

Подавляющее большинство задач, возникающих в таких областях как автоматизация проектирования СБИС, создание систем искусственного интеллекта, распознавания изображений, разработка сетей связи и криптография, относятся к классу комбинаторно-логических. Под комбинаторно-логическими задачами подразумеваются перечислительные и поисковые задачи на конечных множествах, элементами которых служат объекты, представляющие собой комбинации элементов других конечных множеств, разбиения, покрытия, решения систем логических уравнений и т.п. Все эти задачи конечны, т.е. для них существует тривиальный алгоритм перебора по дереву возможных решений, однако число перебираемых при этом вариантов растет экспоненциально с увеличением размерности параметров решаемых задач.

Трудоемкость порождаемых практикой задач постоянно растет, причем не только «количественно» (за счет увеличения размерности параметров), но и «качественно» (за счет увеличения логической и алгоритмической сложности). На практике приходится решать комбинаторно-логические задачи больших размерностей, и возможности решения этих задач для практических параметров сложности находятся на грани возможностей современной вычислительной техники. Становится проблематично решать такие задачи за приемлемое на практике время даже с использованием самых быстродействующих персональных компьютеров.

Параллельная обработка данных при автоматизированном проектировании СБИС получает признание как средство повышения вычислительных возможностей новых инструментов автоматизированного проектирования [1]. Доминирующей архитектурой суперкомпьютерных систем, обеспечивающих параллелизм вычислений, являются кластерные компьютеры, относящиеся к классу МКМД – множественный поток команд – множественный поток данных (MIMD – multiple instruction, multiple data) по классификации Флинна [2]. Кластеры составляют подкласс МКМД – системы с индивидуальной памятью. Архитектура кластерных систем доминирует в рейтинге наиболее производительных ЭВМ. Важнейшим достоинством кластеров является их относительная дешевизна (на единицу пиковой производительности).

В докладе описывается технология разработки программ для кластера, используя инструменты для операционной системы Windows, а также конфигурация программных средств, обеспечивающих подготовку к выполнению и запуску программ на кластере, головная машина которого представляет собой UNIX сервер [3], с помощью удаленного компьютера с Windows. Использование этой технологии позволяет программисту, имеющему навыки работы с инструментами для Windows, использовать эти знания при разработке программ для кластера, работающих под системой Linux [4].

Кластерные компьютеры.

В Объединенном институте проблем информатики Национальной академии наук Беларуси были созданы и используются суперкомпьютеры семейства СКИФ [5], которые имеют архитектуру Beowulf [6] и используют операционную систему Linux.

Для кластеров типа Beowulf, характерна архитектура из автономных вычислительных узлов, объединенных компьютерной сетью и предназначенных для решения одной задачи, как правило, большой вычислительной сложности. На абсолютном большинстве кластеров в качестве операционной системы используется операционная система Linux. Головной (управляющий) узел (front-end node) управляет всем кластером и является файл-сервером для вычислительных узлов. Он также является консолью кластера и шлюзом во внешнюю сеть.

В настоящее время параллельные кластеры типа Beowulf для многих применений далеко превзошли возможности классических суперкомпьютеров, и усовершенствования возможности передачи данных по сети сделали возможными параллельные вычисления, основанные на использовании гетерогенных кластеров и технологии «грид». Такие понятия, как мобильность, предсказуемость, правильность поведения существенно связаны с методологиями разработки программного обеспечения и играют теперь столь же важную роль в параллельном программировании, как ускорение работы программы. Однако все упомянутые качества приобретают значение при условии эффективности параллельного программного обеспечения.

Проблемы разработки параллельных алгоритмов решения давно и интенсивно исследуются. Для разработки конкурентоспособных параллельных программ требуется разработка новых параллельных алгоритмов. Цикл разработки программного обеспечения для параллельных алгоритмов значительно более длинен, чем для последовательных алгоритмов. Это имеет два важных следствия. Прежде всего, программы для параллельных

вычислений являются значительно более дорогостоящими, чем их последовательные аналоги. Этот недостаток усиливается нехваткой мобильности программ для параллельных машин различной архитектуры. Второе следствие более фундаментально и связано с характеристиками самих алгоритмов. Учитывая быстрый темп развития и усовершенствования последовательных алгоритмов для решения задач автоматизированного проектирования, последовательные программы часто выигрывают у параллельных программ из-за трудности разработки последних. Параллельные алгоритмы разрабатываются на основе совершенно иных моделей решения задачи, чем для случая последовательных алгоритмов [7]. Для некоторых задач модель решения изначально параллельна, однако для подавляющего числа других задач такую модель чрезвычайно трудно найти.

Сложно создавать и отлаживать даже последовательные программы для решения комбинаторных задач больших размерностей, а параллелизм вносит свой еще более высокий уровень сложности. В самых простых с виду параллельных программах обнаруживаются иной раз фатальные ошибки. Ограниченность круга разработчиков параллельных алгоритмов можно объяснить также и отсутствием развитой методологии предотвращения или обнаружения ошибок в параллельных программах. Даже отладка параллельных программ требует методов и инструментов, значительно отличающихся от последовательного случая.

Параллельные алгоритмы.

Формализация понятия обычного или *последовательного* алгоритма связана с определением формальной системы, состоящей из формального языка и абстрактной машины, интерпретирующей программы этого языка. Был предложен целый ряд таких машин (машина Тьюринга, машина Поста, нормальные алгоритмы Маркова, рекурсивные функции и др.). Все они являются представителями (потенциально бесконечного) класса «универсальных вычислительных машин».

Различные универсальные вычислительные машины отличаются эффективностью реализации алгоритмов. Более высока эффективность таких вычислительных машин, которые состоят из нескольких процессоров, так как они могут выполнять несколько шагов вычисления одновременно. Программы для этих машин задаются параллельными алгоритмами. Эти алгоритмы с функциональной точки зрения ничем не отличаются от последовательных алгоритмов, и задачу разработки параллельной программы можно рассматривать как задачу повышения эффективности выполнения последовательного алгоритма путем распараллеливания. Проблема распараллеливания ставится как задача оптимизации, то есть как задача выбора наилучшего по эффективности параллельного представления последовательного алгоритма.

Абстрактный механизм для формализации параллельных алгоритмов является расширением абстрактных машин последовательных алгоритмов механизмом взаимодействия параллельных ветвей. Однако в параллельных вычислениях отсутствует универсальная абстрактная параллельная машина. Отсутствие такой машины делает невозможным решение задачи распараллеливания, независимое от архитектуры параллельной системы.

В программах обработки данных основным источником массового параллелизма являются программные циклы. Распараллеливанию таких программ посвящено большое количество работ, результаты которых нашли применение в распараллеливающих компиляторах. Для этих программ характерна возможность статического распределения и таких ресурсов, как процессоры. Для того чтобы этот метод работал, длины циклов должны быть известны на стадии компиляции, т.е. должны являться константами. Программы для решения логико-комбинаторных задач не могут быть распараллелены таким методом потому, что структура и длина циклов в этих программах изменяется в ходе вычислений.

Разработка параллельных программ для кластера.

Для эффективного использования ресурсов кластера необходимо обеспечить равномерную загрузку процессоров, используемых параллельной программой, это в свою очередь означает, что все ветви параллельной программы должны выполнить примерно одинаковый объем вычислительной работы. Основной моделью параллельного программирования на кластере является «модель передачи сообщений», которая обеспечивается интерфейсом Message Passing Interface (MPI) [8]. Процедурный язык программирования и библиотеки стандарта MPI являются средствами «низкого уровня» применительно к задаче параллельного программирования, подобно тому, как язык ассемблера – низкоуровневое средство применительно к задаче кодирования последовательных вычислительных алгоритмов.

Программный интерфейс MPI представляет собой описание точного формата вызовов подпрограмм и смысла этих подпрограмм, составляющих библиотеку функций для использования при программировании на языках С или ФОРТРАН. Стандарт исключает какое-либо скрытое (не включенное в предоставленный программисту интерфейс) взаимодействие ветви параллельного процесса со средой MPI. Это позволяет эффективно переносить программы с одной реализации MPI на другую. Базовой реализацией MPI является библиотека MPICH (Message Passing Interface CHameleon). При этом широко распространены коммерческие продукты, ориентированные на аппаратные особенности архитектуры кластера.

Исполнимый код MPI программы перед ее исполнением копируется в память каждого из процессов группы узлов кластера. Все участки MPI-программы делятся на два типа: общие, исполняемые всеми процессорами, и выделенные, исполняемые только определенным процессом или группой процессов. Все узлы кластера включаются в работу одновременно и работают параллельно. При этом каждый из процессов должен «распознавать» и выполнять только свои и общие участки общей программы, обмениваясь сообщениями с другими процессами при необходимости.

Технология разработки программы для кластера.

В операционной системе Linux для кластера имеется среда для разработки программ на C++, которая предоставляет программисту удобства, похожие на возможности среды MSVC в Windows, однако работа в этой среде значительно отличается от работы в среде MSVC. Компилятор C++ на кластере управляется средствами командной строки. Такой компилятор есть во всех системах UNIX. Но между разными машинами, даже имеющими один и тот же тип UNIX, не существует совместимости программ на уровне двоичных кодов. Для машин с системой Linux существует совместимость программ на уровне двоичных кодов при одинаковой версии ядра операционной системы. Таким образом, установка любой программы (за исключением программ, которые выполняются интерпретаторами) в UNIX обычно требует компиляции ее исходного кода.

Более привлекательна стратегия разработки программы на основе концепции переносимого кода, которая обеспечивает независимость разрабатываемой программы от среды разработки. Для тех, кто разрабатывал программы для Windows на C++ и хотел бы пользоваться привычными технологиями и инструментами при разработке программы для кластера, довольно удобна предлагаемой стратегии, которая позволяет в какой-то мере использовать среду MSVC для разработки параллельной программы: в среде MSVC разрабатывается последовательный прототип параллельной программы с учетом требования переносимости кода. Переносимость кода означает использование консольной программы в MSVC. Разработанный программный код преобразуется в параллельную программу, которая может быть выполнена на кластере. Для этого текст программы должен быть откомпилирован в среде Linux.

Такую технологию разработки параллельной программы можно реализовать, если использовать компьютер, работающий под Windows, как удаленный терминал кластерного

компьютера с помощью свободно доступной консольной программы PuTTY. Один из существенных недостатков такого метода доступа к кластеру с компьютера состоит в том, что в этом случае приходиться ограничиваться теми ресурсами Linux, которые доступны через консоль. Поэтому через удаленный терминал программы PuTTY в принципе нельзя использовать программы Linux с графическим интерфейсом. С другой стороны, использование такой среды для разработки программ под Linux позволяет вникать в как можно меньшее число технических деталей, связанных с управлением операционной системой Linux при редактировании, компиляции и запуске программы на выполнение, а сосредоточиться на разработке самого параллельного алгоритма решения задачи.

Компиляция программы.

Задание на компиляцию программы с использование компилятора C++ Linux представляет собой управляющий файл для программы *make*. Таким образом, для компиляции программы кроме ее текстов на C++ нужно иметь файл задания для *make*. Среда MSVC позволяет экспортировать файл проекта в формате *make*, однако файл, экспортированный из MSVC для управления компиляцией на Linux, требует такой ручной переделки, что проще его написать заново с нуля.

Можно компилировать программу и без задания для *make*, вызывая компилятор C++ непосредственно из командной строки, но в этом случае текст программы должен быть представлен в виде одного файла, а все остальные части текста должны подключаться явно с помощью директив *#include*. Использование *make* позволяет ускорить компиляцию программы, которая состоит из нескольких файлов, и выполнять перекомпиляцию только измененных файлов с исходным текстом.

Формат файла для *make* довольно сложен, и при построении задания нужно указать все файлы, которые потребуются при компиляции программы, и их связи; нужно предусмотреть управление каждым из используемых инструментов (например, редактором связей кроме компилятора). Однако существуют программы, позволяющие сгенерировать управляющий файл автоматически, например, свободно доступная программа *genmake* (http://www.robertnz.net/genmake.htm).

Проект, для которого генерируется управляющий файл для *make*, должен быть доработан: 1) в каждый *.h* файл проекта нужно включаются в формате комментария специальные директивы, в которых указываются имена *.cpp* файлов, содержащих определения объектов, декларированных в соответствующем *.h* файле, 2) для *genmake* указывается название *.cpp* файла, который содержит функцию *main* программы и который просматривается программой *genmake* для нахождения директив *#include* для включения *.h* файлов и поиска .cpp файлов, содержащих определения объектов, декларированных в соответствующем *.h* файле. Таким образом, разыскиваются все требуемые файлы и формируются зависимости между ними. К сожалению, сгенерированный *genmake* файл все-таки требует небольшой ручной доработки.

Еще одной проблемой является использование русского языка: существует несколько кодировок для русского языка – Windows 1251, KOI-8r, ISO 8859-5 и др. По умолчанию кодовой страницей русского языка для Linux является KOI-8r, а для Windows – это Windows 1251. Различие в стандартах кодировки ведет к тому, что тексты программ, подготовленные для компиляции в Windows, будут неправильно выглядеть при просмотре на Linux. Но проблема не только в просмотре текста программы, через консоль очень трудно выполнить переключение раскладки клавиатуры на русский язык при редактировании. По этой причине редактирование текстов программ удобнее выполнять на удаленном компьютере пользователя с Windows. Подходящим редактором является среда MSVC. Но для правильной работы с русским языком с учетом перехода в среду Linux редактор должен обеспечивать возможность преобразования кодировки Windows 1251 в KOI-8r и наоборот. Для этих целей удобен текстовый редактор *Aditor*, который поддерживает почти все кодировки русского алфавита (KOI, Win, DOS, ISO, Mac). В этом редакторе можно

просматривать и редактировать файлы в любой из этих кодировок (и при открытии файла его кодировка определяется автоматически), можно также переводить файлы из одной кодировки в другую. Кроме того, в этом редакторе подсвечивается цветом синтаксис C++.

Технология работы с кластерным компьютером.

Компьютер с ОС Windows должен быть подключен к той же сети, что и головная машина кластера. Для интерактивной работы с кластером и обмена файлами требуется использовать протокол SSH с шифрованием информации (Secure SHell) [9].

Для работы с кластерным компьютером создается три одновременно запущенные в ОС Windows процесса: редактора *Aditor*, программы транспортировки файлов *SecureFX* и программа *PuTTY* удаленного терминала для управления кластером, в которой на кластере запускается файловый менеджер *Midnight Commander*. Этот файловый менеджер входит в комплект поставки Linux и имеет текстовый интерфейс типа Norton Commander, разработанный для операционных систем типа Unix. Программа SecureFX обеспечивает зашифрованную передачу файлов с возможностями настройки конфигурации и протоколов передачи, которая включает в себя утилиту для передачи файлов по протоколу SSH из командной строки.

Основой организации работы с программой для кластера является каталог на компьютере с ОС Windows с текстами программы. Эти тексты отличаются от текстов исходного прототипа для Windows не только параллельностью, но также тем, что перекодированы в KOI-8r (с помощью программы Aditor) и в *.h* файлы включены директивы *genmake*. Создание и заполнение такого каталога является первым этапом разработки программы для кластерного компьютера. Затем с помощью программы *genmake* строится и подправляется *make*-файл. Далее полученный каталог с данными дублируется в каталог на головной машине кластера. Для этого данные из каталога на машине с ОС Windows транспортируются в каталог на головной машине кластере (с помощью программы транспортировки файлов SecureFX). На этом подготовительная работа для компиляции и запуска программы заканчивается.

Каталог на кластере служит зеркалом каталога на компьютере разработчика. Файлы, с которыми идет работа, удобно держать открытыми в программе Aditor. Для компиляции и выполнения программы нужно запустить программу PuTTY и в ней запустить файловый менеджер Midnight Commander, набирая *mc*. Если при компиляции возникли ошибки, то номера строк программы и диагностику ошибок можно посмотреть под панелями Midnight Commander. Используя открытые в редакторе Aditor файлы и эти номера, можно найти соответствующие строки кода программы и устранить ошибки. После редактирования кода нужно сохранить исправления (не закрывая Aditor) и с помощью SecureFX транспортировать исправленные файлы в каталог на кластере. После обновления каталога на кластере нужно повторить компиляцию.

Если компиляция прошла успешно, двоичный файл программы появляется на панели Midnight Commander отмеченным звездочкой, и его можно выполнить. Запуск программы на выполнение выполняется теми же способами, что и в «Нортоне» [10]: нужно в поле командной строки программы mc набрать ее имя, нужные параметры и нажать ввод. Следует учитывать, что исполняемые файлы программ в двоичном виде в Linux не имеют расширения. Пока так запущенная программа работает, пользователь ничего делать не может, должен ждать ее завершения. Кроме обычного режима запуска существует еще режим фонового запуска, который позволяет освободить консоль для выполнения других работ. Что запустить программу в фоновом режиме, надо после имени файла поставить символ «&». После завершения программы результаты работы можно посмотреть под панелями Midnight Commander.

Заключение.

Предлагаемая конфигурация программных средств для подготовки и выполнения параллельных программ для кластера не является единственной из возможных. Однако использование предлагаемой среды позволяет программисту, имеющему навыки работы с инструментами для Windows, использовать эти знания при разработке программ для кластера. Это, конечно, дается ценой отказа от использования некоторых средств, предоставляемых рабочей системой на основе Linux. Основной недостаток предлагаемого подхода состоит в невозможности использования инструментов разработки программ Linux с графическим интерфейсом. Рекомендации для пользователей кластера, работающих на рабочих компьютерах с Linux, приведены в [4].

Выполнено практическое применение предлагаемой конфигурации для разработки нескольких параллельных программ [11], последовательные прототипы которых были построены в среде MSVC, позволило добиться того, что перенос последовательного алгоритма в Linux не требовал умения работать в Linux, и основные трудозатраты состояли в разработке и отладке параллельного алгоритма, а не в овладении комплексом инструментальных средств.

Список использованных источников

[1]. Banerjee, P. Parallel Algorithms For VLSI Computer-Aided Design / P. Banerjee. – Prentice Hall, Englewoods Cliffs, NJ, 1994.

[2]. Flynn, M. J. Some computer organizations and their effectiveness / M.J. Flynn // IEEE Transactions on Computers. – 1972. – 21 (9). – P. 948–960.

[3]. Робачевский, А. М., Операционная система UNIX / А.М. Робачевский, С.А. Немнюгин, О.Л. Стесик. – СПб.: БХВ-Петербург, 2-е изд., 2010. – 656 с.

[4]. Галактионов, В.В. Руководство для пользователей LINUX кластера ЛИТ ОИЯИ / В.В. Галактионов, Т.М. Голоскокова, Н.И. Громова, и др. – Дубна, 2004.

[5]. Абрамов, С.М. Принципы построения суперкомпьютеров семейства «СКИФ» и их реализация / С.М. Абрамов, Н.Н. Парамонов, В.В. Анищенко, С.В. Абламейко // Информатика. – 2004. – № 1. – С. 89–106.

[6]. Sterling, T. Beowulf: A Parallel Workstation for Scientific Computation / T. Sterling, D. Becker, D. Savarese, et al. // Proc. of Intern. Conf. on Parallel Processing. – Oconomowoc, 1995. – P. 11–14.

[7]. Черемисинов, Д.И. Анализ и преобразование структурных описаний СБИС / Д.И. Черемисинов. – Минск: Беларус. навука, 2006. – 275 с.

[8]. Message Passing Interface Forum. MPI: A Message-Passing Interface standard, version 1.1. // [Электронный ресурс], Available at: http://www.mpi-forum.org/docs (date of access: 10.03.2022).

[9]. Сервер OpenSSH // [Электронный ресурс], Available at: https://help.ubuntu.ru/wiki/руководство_по_ubuntu_server/удаленное_администрирование/openssh_server. – Date of access: 10.03.2022.

[10]. Фигурнов, В.Э. IBM PC для пользователя. От начинающего до опытного / В.Э. Фигурнов. – М.: ИНФРА-М, 2002. – 640 с.

[11]. Черемисинов, Д.И. Использование параллельных вычислений при автоматизированном проектировании СБИС / Д.И. Черемисинов, Л.Д. Черемисинова // Проблемы разработки перспективных микро- и наноэлектронных систем. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского, М.: ИППМ РАН, 2016. Часть I. – С. 32-39.

SOFTWARE ENVIRONMENT FOR WORKING WITH A CLUSTER COMPUTER FROM OC WINDOWS

D.I. CHEREMISINOV
Leading researcher of UIIP of NAS of Belarus, candidate of technical sciences, associate professor

L.D. CHEREMISINOVA
Principal researcher of UIIP of NAS of Belarus, doctor of technical sciences, professor

United Institute of Informatics Problems of National Academy of Sciences of Belarus, Republic of Belarus
E-mail: {cher, cld}@newman.bas-net.by

Abstract. The problem of developing and execution of programs for multiprocessor cluster-type systems is considered. When developing programs for a cluster computer, a technology based on the use of a remote terminal is used. We consider the situation when such a remote terminal is a computer with the Windows operating system. A set of program tools is proposed that allows you to perform the tasks of editing text, compiling a program, and running the program on a cluster. The advantage of the proposed method of preparing a program for execution is that it allows a programmer with experience in working with tools for Windows to use this knowledge when developing programs for a cluster computer operating in a UNIX environment.

Keywords: concurrent computing, cluster computer, programming technology.