Лабораторная работа №7

Параллельное программирование

Цель: познакомиться с особенностями параллельного программирования. Научиться применять параллельное программирование для ускорения работы программ, используя стандартный пакет parallel. Решить задания в соответствующем стиле программирования. Составить отчет.

Теоретическая часть

Существует несколько методов для запуска кода параллельно. В языке программирования R, для распараллеливания выполнения кода можно использовать пакета из ядра языка parallel, так и другие пакеты, например foreach и future.apply.

Методы разделения задачи для решения в параллельном стиле:

1. Разделение на задачи. Каждую из задач можно выполнить независимо друг от друга. В данных задачах используются разные данные.

2. Разделение по данным. Данный метод встречается часто для обработки больших данных.

Примером решения задачи разделения по данным может быть нахождения суммы элементов ряда: 1 + 2 + 3 + … + 100. В данном случае ряд можно разбить на 4 части и посчитать их параллельно и независимо друг от друга:

```
sum(1:25) + sum(26:50) + sum(51:75) + sum(76:100)
```

С точки зрения использования памяти существует два принципиальных подхода для решения задач параллельных вычислений:

1. Использование общей памяти (рис. 1)

2. Использование разделенной памяти (рис. 2)

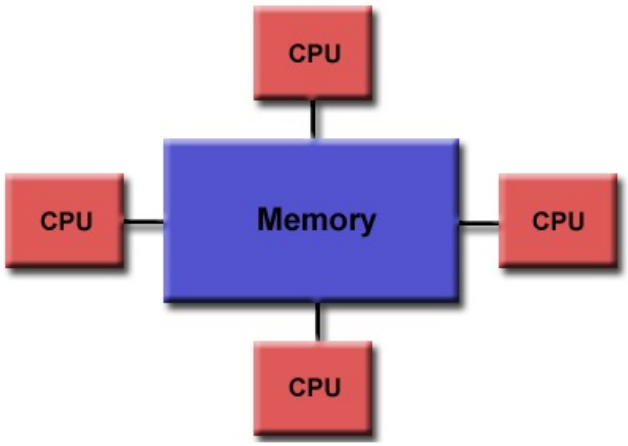

Рисунок 1 – Общая память

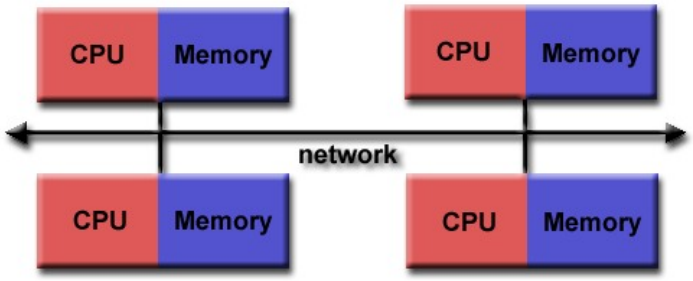

Рисунок 2 – Разделенная память

Существует два варианта реализации парадигмы параллельного программирования: модель «Master-worker» (мастер-работник), также встречается в названии «Master-slave» и модель Map-reduce (уменьшение карты).

На Map-шаге происходит предварительная обработка входных данных. Для этого один из компьютеров (называемый главным узлом – master node) получает входные данные задачи, разделяет их на части и передает другим компьютерам (рабочим узлам — worker node) для предварительной обработки.

На Reduce-шаге происходит свертка предварительно обработанных данных. Главный узел получает ответы от рабочих узлов и на их основе формирует результат – решение задачи, которая изначально формулировалась.

Модель Master-worker наиболее типичная для простых параллельных вычислений и имеет вид, представленный на рисунке 3.

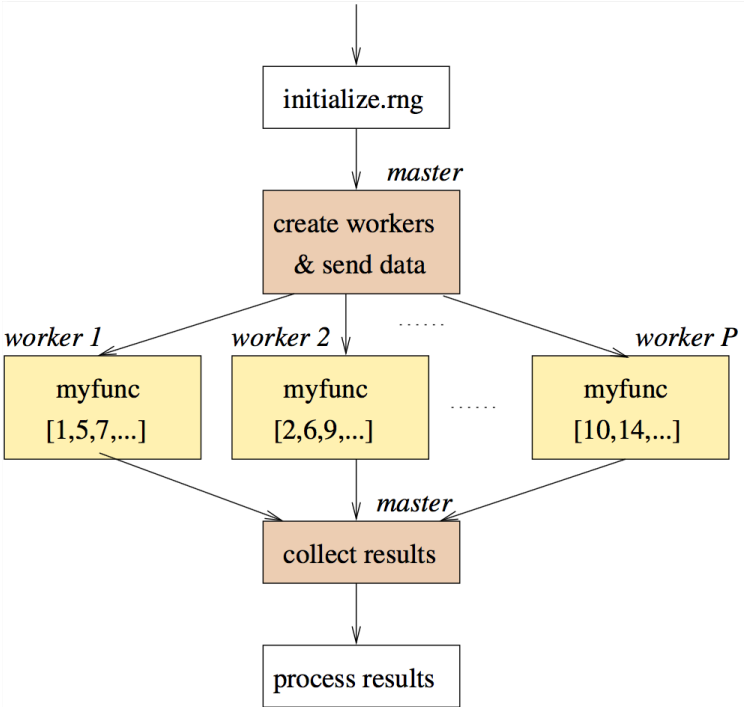

Рисунок 3 – Модель Master-worker

В самом простом виде реализация этой модели возможна с использованием пакета parallel, входящего в состав базового ядра языка R.

```r
# Подключение пакета, определение количества ядер
library(parallel)
ncores <- detectCores(logical = FALSE)
n <- ncores:1

# Создание кластера
cl <- makeCluster(ncores)

# Использование кластера с clusterApply для вызова функции rnorm для
# каждого n параллельно с параметрами mean = 10 и sd = 2
clusterApply(cl, n, rnorm, mean = 10, sd = 2)

# Остановка кластера
stopCluster(cl)
```

Рисунок 4 – Пример кода создания и использования вычислительного кластера

Практическая часть

Задание 1.

Используя заранее подготовленные функции визуализируйте сведения о наиболее часто встречающихся словах из книг Джейн Остин по буквам английского алфавита. Книги, необходимые для анализа, находятся в пакете janeaustenr. Также для работы потребуется пакет stringr. После установки пакетов добавьте следующие функции:

```
extract_words <- function(book_name) {
  text <- subset(austen_books(), book == book_name)$text
  str_extract_all(text, boundary("word")) %>% unlist %>% tolower
}

janeausten_words <- function() {
  books <- austen_books()$book %>% unique %>% as.character
  words <- sapply(books, extract_words) %>% unlist
  words
}

max_frequency <- function(letter, words, min_length = 1) {
  w <- select_words(letter, words = words, min_length = min_length)
  frequency <- table(w)
  frequency[which.max(frequency)]
}
select_words <- function(letter, words, min_length = 1) {
  min_length_words <- words[nchar(words) >= min_length]
  grep(paste0("^", letter), min_length_words, value = TRUE)
}
```

Рисунок 5 – Функции необходимые для решения задания

Для решения задачи необходимо с помощью функции janeausten_words() создать новый объект – вектор слов. Далее, используя одну из функций семейства apply, и заранее созданную функцию max_frequency создать именованный вектор, содержащий значение максимально встречающих слов английского алфавита, длиной не менее 5 букв.

Полезной для работы будет использование встроенной переменной letters, содержащей строчные буквы английского алфавита. Для визуализации используйте функцию barplot() с аргументом las = 2. В случае полностью правильного выполнения задания будет представлен график как на рисунке 6.

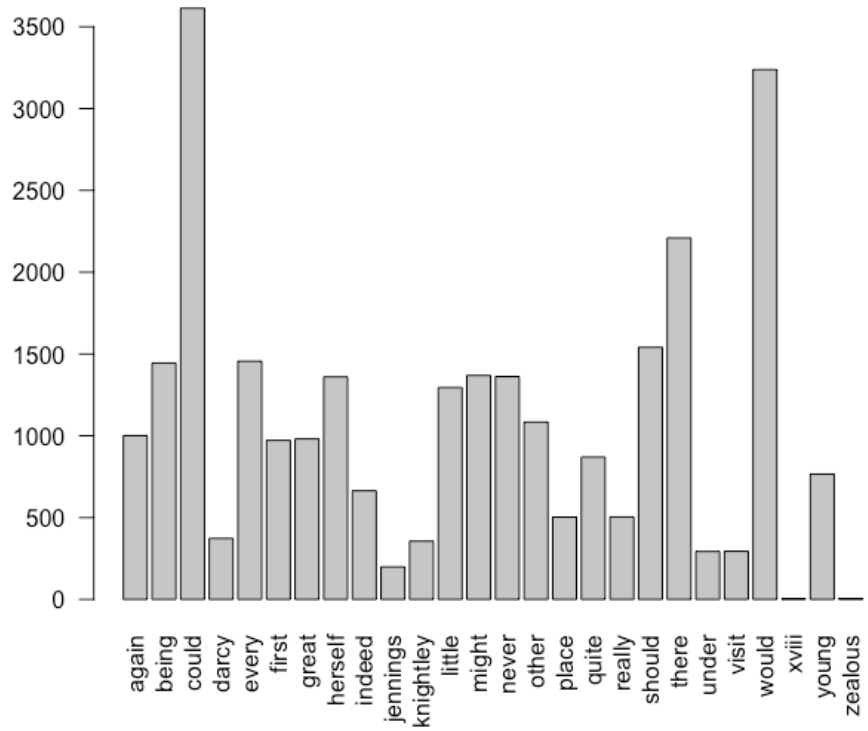

Рисунок 6 – Решение задания 1

Задание 2.

Распараллельте фрагмент кода, представленный ниже, используя вычислительный кластер:

```
for(iter in seq_len(50))
    result[iter] <- mean_of_rnorm(10000)
```

Для решения подгрузите функцию

```
mean_of_rnorm <- function(n) {
  random_numbers <- rnorm(n)
  mean(random_numbers)
}
```