Meta

Examinar las diferencias en los tiempos de ejecución de los diferentes ordenamientos en función al número de núcleos asignados al clúster.

Código

```
> library(ggplot2)
> source('~/GitHub/Simulacion/Simulacion/P3/Obt_primos.R')
>
> primo <- function(n) {
+   if (n == 1 || n == 2) {
+     return(TRUE)
+   }
+   if (n %% 2 == 0) {
+     return(FALSE)
+   }
+   for (i in seq(3, max(3, ceiling(sqrt(n))), 2)) {
+     if ((n %% i) == 0) {
+       return(FALSE)
+     }
+   }
+   return(TRUE)
+ }
>
> desde <- 100
> hasta <-  300
> original <- desde:hasta
> invertido <- hasta:desde
> prgr<-tail(primos,hasta-desde)
> prpq<-primos[hasta-desde]
>
> replicas <- 30
> suppressMessages(library(doParallel))
> resultados<-data.frame()
>
> for (t in 1:(detectCores()-1)){
+   registerDoParallel(makeCluster(t))
+   ot <-  numeric()
+   it <-  numeric()
+   at <-  numeric()
+   gt<-numeric()
+   pt<-numeric()
+
+   for (r in 1:replicas) {
+     ot <- c(ot, system.time(foreach(n = original, .combine=c) %dopar% p
rimo(n))[3])
+     it <- c(it, system.time(foreach(n = invertido, .combine=c) %dopar%
primo(n))[3])
+     at <- c(at, system.time(foreach(n = sample(original), .combine=c) %
dopar% primo(n))[3])
+     gt <- c(gt, system.time(foreach(n = prgr, .combine=c) %dopar% primo
(n))[3])
```

```
+     pt <- c(pt, system.time(foreach(n = prpq, .combine=c) %dopar% primo
(n))[3])
+
+
+ }
+
+   stopImplicitCluster()
+   ot <-  matrix(ot,byrow=TRUE)
+   it <-  matrix(it,byrow=TRUE)
+   at <-  matrix(at,byrow=TRUE)
+   gt<-matrix(gt,byrow=TRUE)
+   pt<-matrix(pt,byrow=TRUE)
+
+
+   ot<-cbind(ot,1,t)
+   it<-cbind(it,2,t)
+   at<-cbind(at,3,t)
+   gt<-cbind(gt,4,t)
+   pt<-cbind(pt,5,t)
+
+   resultados<-rbind(resultados,ot,it,at,gt,pt)
+   resultados<-data.frame(resultados)
+
+
+ }
Warning messages:
1: In .Internal(gc(verbose, reset)) :
  closing unused connection 4 (<-AngelesMttz:11399)
2: In .Internal(gc(verbose, reset)) :
  closing unused connection 3 (<-AngelesMttz:11399)
3: In .Internal(gc(verbose, reset)) :
  closing unused connection 5 (<-AngelesMttz:11399)
4: In .Internal(gc(verbose, reset)) :
  closing unused connection 4 (<-AngelesMttz:11399)
> colnames(resultados)<-c("tiempo","Orden","Nucleos")
> resultados$Nucleos <- factor(resultados$Nucleos,
+                  labels = c("Uno", "Dos", "Tres"))
> resultados$Orden <- factor(resultados$Orden,
+                            labels = c("Original", "Invertido", "Aleat
orio","Primos grandes","Primos pequeños"))
>
> png("Variacion_en_nucleos.png")
> ggplot() +
+   geom_boxplot(data=resultados, aes(x = Nucleos, y=tiempo,fill =Orden))
+
+   scale_y_continuous(name="Tiempo de ejecucion") +
+   scale_x_discrete(name="Nucleos")
> dev.off()
```

Desarrollo del código

El código R presentado en la sección anterior es resultado de modificaciones al código R utilizado como ejemplo en clase [1]. Para la creación del gráfico ggplot se utilizó un código ejemplo encontrado en línea [2]. Estas modificaciones, presentadas en celeste, fueron realizadas con los siguientes objetivos:

1. Cargar la librería “ggplot2” para obtener gráficos más atractivos que un *boxplot* básico.
2. Utilizar el script “Obt_primos.R” para obtener el vector “primos”, el cual contiene únicamente números primos dentro del rango de 1 a 30,000; dichos elementos están ordenados de manera ascendente.
3. Crear dos ordenamientos contrastantes para apreciar una drástica variación en el tiempo de ejecución. El vector “prgr” que contiene los últimos elementos del vector “primos”, es decir, contiene los números primos más grandes dentro de este intervalo de valores. El vector “prpq”, contrario al vector “prgr”, contiene los primeros valores del mencionado vector, es decir, incluye los números primos más pequeños para este rango de valores.
4. Guardar los resultados “temporales” dentro de una variable “resultados”
5. Realizar el cambio en el número de núcleos asignados en el clúster en función a la variable “t”, la cual va de uno a detectCores()-1.
6. Señalar un conjunto vació donde se guardarán los resultados para cada tipo de ordenamiento.
7. Medir el tiempo de ejecución de los nuevos vectores con números primos.
8. Crear un matriz con los siguientes elementos por columnas: tiempo de ejecución, nombre del ordenamiento y número de núcleos utilizados.
9. Añadir los nuevos valores en cada ciclo del cambio en número de núcleos.
10. Presentar los “resultados” en un data.frame()
11. Asignar nombres a las columnas del data.frame “resultados”.
12. Proporcionar un nombre significativo para los valores de “t” el cual es el número de núcleos y a los valores “Orden” de acuerdo a las etiquetas señaladas para cada conjunto de datos.
13. Creación de un ggplot que muestra el tiempo de ejecución en función al número de núcleos para cada tipo de ordenamiento.

Nota: Este código muestra mensajes de advertencia (en color marrón) que indican conexiones sin usar

Resultados

En la gráfica 1 se muestran los resultados obtenidos para el código anterior, esta gráfica es presentada en función al número de núcleos (eje x) y el tiempo de ejecución (eje y). Si se observa el valor de la mediana para cada “caja” dentro del gráfico “caja-bigote”, es posible apreciar una leve disminución del tiempo cuando se realiza el paralelismo con dos núcleos en lugar de uno, mientras que al agregar un núcleo más para esta tarea, es decir, que se realice con tres núcleos en lugar de dos, existe una diferencia poco significativa. De estos resultados se puede decir que aumentar el número de núcleos para trabajar en paralelo, no disminuye, en todos los casos, el tiempo de ejecución.

El tiempo de ejecución más bajo se logró en el ordenamiento que incluía únicamente primos pequeños, esto se puede debe a que los cálculos para números pequeños implican menos factores entre los cuales realizar la función “primo”, logrando el procesamiento de los datos

casi de manera inmediata; por lo cual se puede afirmar que si lo que se desea es disminuir el tiempo de ejecución drásticamente, es aconsejable simplificar los datos de entrada para la tarea asignada.

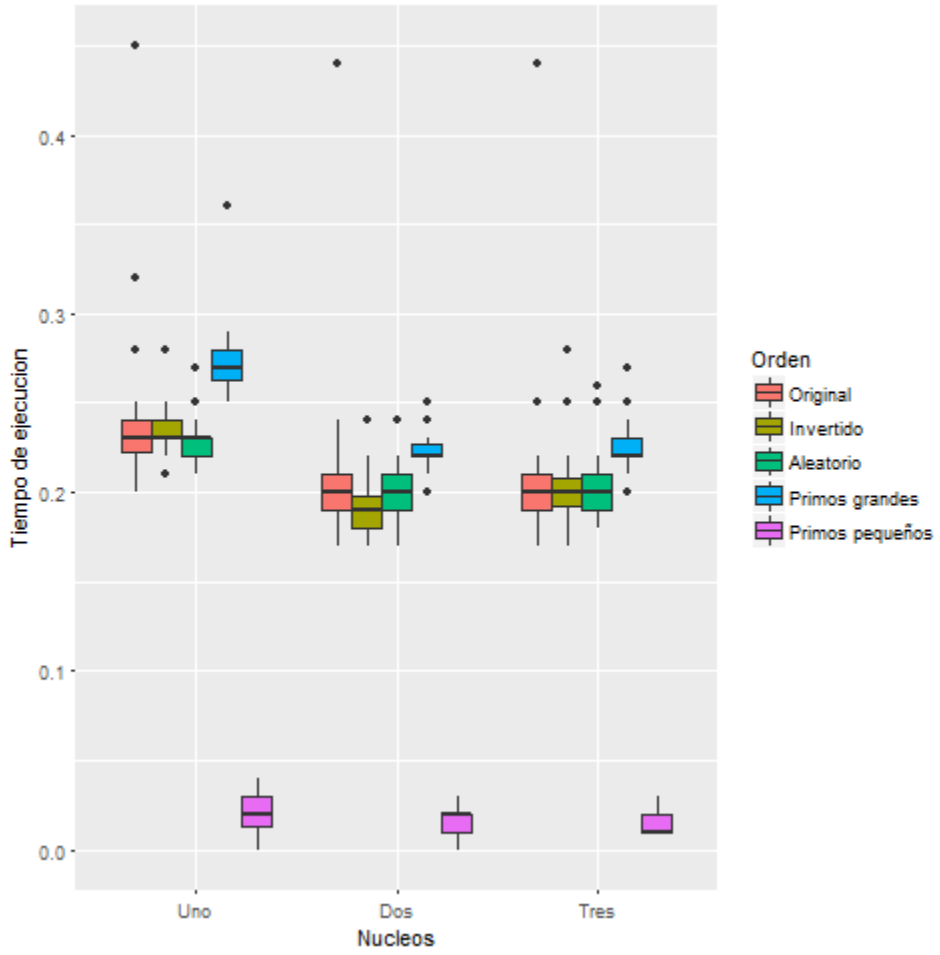

Figura 1. Variación en el tiempo de ejecución de diferentes ordenamientos en función al número de núcleos designados al clúster.

Referencias

[1] Schaeffer, S. (2017). P2 — R paralelo — Schaeffer. [online] Elisa.dyndns-web.com. Available at: http://elisa.dyndns-web.com/teaching/comp/par/p3.html [Accessed 28 Aug. 2017].

[2] Artiles,A(2016). Creating plots in R using ggplot2 - part 10: boxplots. [Online] Available at: http://t-redactyl.io/blog/2016/04/creating-plots-in-r-using-ggplot2-part-10-boxplots.html [Accesed 28 Aug.2017]