Reporte práctica 3
Teoría de colas

21 de agosto de 2017

1. Introducción

En esta práctica se desea estudiar los efectos de la calendarización de tareas/trabajos sobre los núcleos del procesador. Una tarea consiste en determinar si un número n es primo o no. En términos generales, se espera que entre más grande sea n sea mayor el tiempo requerido para determinar si es primo (o en contexto, mayor el tiempo de procesamiento de la tarea). La forma en que se programan las tareas depende de la librería `doParallel` de R, pero se puede influir con el orden en que se le entreguen las tareas para que las procese. Si calculamos el tiempo en el que todas las tareas son procesadas, estaremos midiendo el efecto de la calendarización de las tareas; obviamente éste tiempo dependerá de la cantidad de núcleos que se utilicen.

En este documento se presentan el análisis de los resultados obtenidos, las pruebas estadísticas que sustentan las conclusiones realizadas y algunas explicaciones intuitivas con apoyo gráfico. En resumen, tanto la tarea como ambos retos se encuentran mezclados (por la naturaleza de la práctica) para lograr un mejor desenvolvimiento.

2. Experimento

En el experimento se utilizan todos los enteros $n \in [100, 10000]$ y se definen tres ordenes diferentes: ascendente, descendiente y aleatorio. Además, se utilizan como medidas de comparación un caso ideal en donde todos los número son pares y uno nadir en donde se colocan múltiples copias de 9973, el cual es el número primo más grande en el intervalo de estudio. El tiempo de proceso de los números pares es mínimo pues se revisa la paridad del número antes de buscar sus otros divisores; como no tiene importancia cual número sea (mientras sea par), se consideran múltiples copias del número 4. Aunque estos dos últimos no son ordenamientos, los llamaremos así por facilidad. Por último, la cantidad de núcleos utilizados varia de uno a siete y se realizaron treinta réplicas por cada par (orden,núcleos) para medir la variabilidad.

3. Resultados

Los resultados del experimento planteado se ilustran el la Figura 3.1. Particularmente, los diagramas de bigotes correspondientes al número de núcleos utilizados se muestran en las subfiguras (3.1a-3.1g), hay cinco cajas de bigotes una para cada orden de entrega de las tareas. La subfigura (3.1h) muestra la gráfica de las medianas correspondientes a cada par (orden,núcleos).

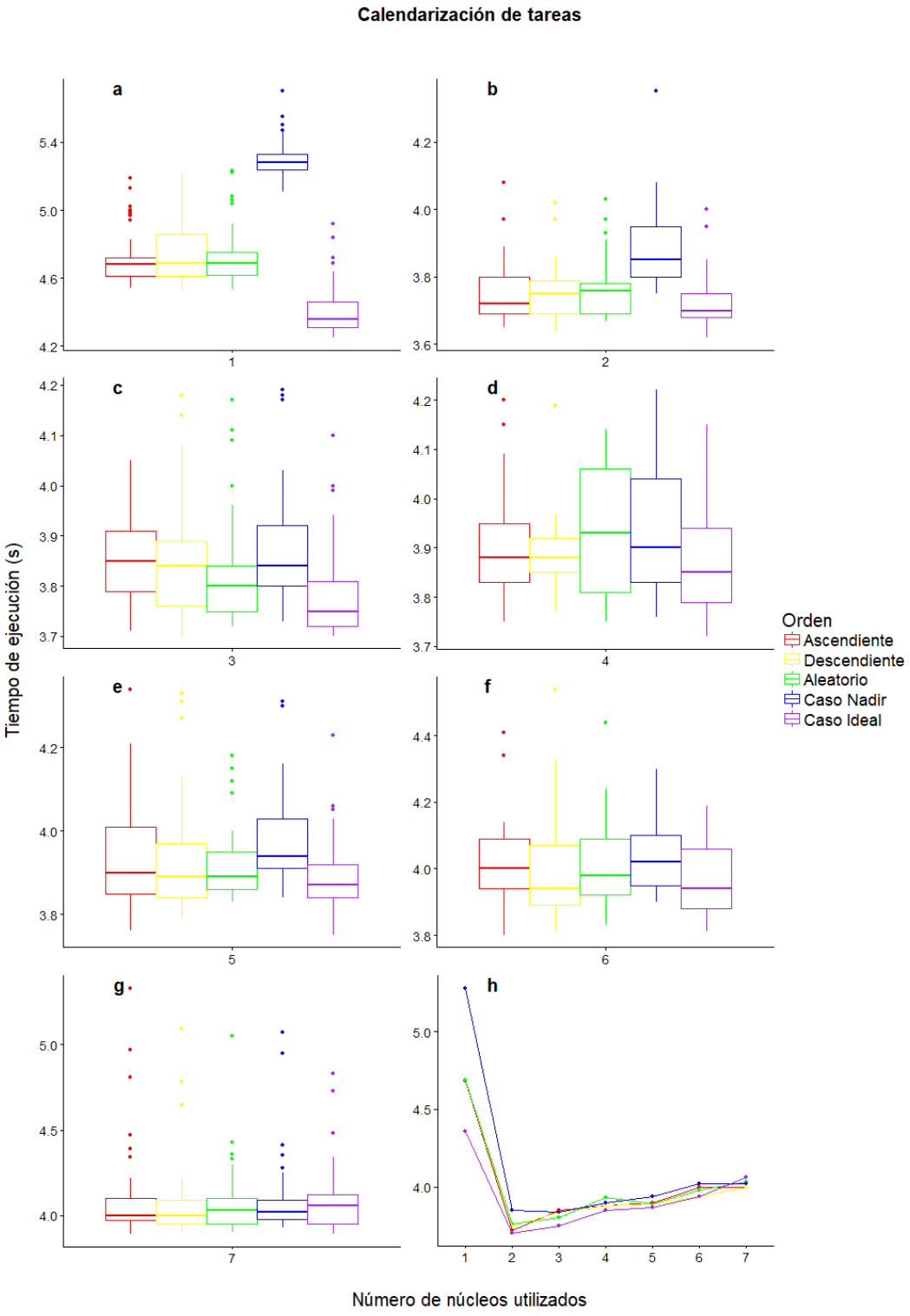


Figura 3.1: Efecto de la calendarización de tareas en dependencia del orden de entrega y de los núcleos utilizados.

De forma intuitiva uno esperaría que entre mas núcleos se apliquen para desarrollar las tareas, más rápido se terminaría de procesarlas. Sin embargo; puede apreciar que nuestra intuición es errónea. Note como el tiempo de proceso disminuye notablemente de resolver las tareas secuencialmente (con un sólo núcleo) a utilizar dos núcleos (véase Figura 3.1h), pero a medida que los núcleos disponibles aumentan lo hace también el tiempo de proceso. Por supuesto hay una clara diferencia entre una aplicación secuencial a una paralela. Una prueba estadistica de Kruskal y Wallis con 99.9% de confianza indica que el número utilizado de núcleos es significativo. Más aún, una prueba de Dunn, con la misma significancia, indica que existe diferencia significativa entre cada par de niveles de este factor. Los resultados de ésta última prueba se muestran enseguida, la columna `P.adj` muestra el valor p de las pruebas entre pares:

	Comparison	Z	P.unadj	P.adj
1	1 - 2	88.861467	0.000000e+00	0.000000e+00
2	1 - 3	75.121784	0.000000e+00	0.000000e+00
3	2 - 3	-13.739683	5.873263e-43	6.491501e-43
4	1 - 4	59.794935	0.000000e+00	0.000000e+00
5	2 - 4	-29.066532	9.512706e-186	1.664723e-185
6	3 - 4	-15.326849	5.059389e-53	5.902621e-53
7	1 - 5	55.900789	0.000000e+00	0.000000e+00
8	2 - 5	-32.960678	2.974559e-238	6.246573e-238
9	3 - 5	-19.220995	2.469889e-82	3.241730e-82
10	4 - 5	-3.894146	9.854546e-05	9.854546e-05
11	1 - 6	40.429585	0.000000e+00	0.000000e+00
12	2 - 6	-48.431881	0.000000e+00	0.000000e+00
13	3 - 6	-34.692199	1.032952e-263	2.410221e-263
14	4 - 6	-19.365349	1.513248e-83	2.118548e-83
15	5 - 6	-15.471204	5.428236e-54	6.705468e-54
16	1 - 7	31.266005	1.353005e-214	2.583009e-214
17	2 - 7	-57.595461	0.000000e+00	0.000000e+00
18	3 - 7	-43.855779	0.000000e+00	0.000000e+00
19	4 - 7	-28.528930	5.128876e-179	8.285107e-179
20	5 - 7	-24.634784	5.357621e-134	8.036431e-134
21	6 - 7	-9.163580	5.020349e-20	5.271367e-20

Pero... ¿qué pasa con la forma en que son entregadas las tareas? ¿esto afecta al tiempo de procesamiento? La respuesta es si, en las subfiguras (3.1fa-3.1c) puede apreciarse notablemente la diferencia en los tiempos de proceso de los casos ideal(morado) y nadir(azul) contra los tres ordenes propuestos. Pero formalmente una prueba estadistica de Kruskal y Wallis muestra con una significancia de 0.001 que el orden afecta el tiempo de proceso. Además, una prueba de Dunn indica que podemos agrupar los ordenes en tres grupos: el caso ideal, el caso nadir y los restantes. Los resultados de la prueba que no tienen significancia estadística aparecen a continuación:

	Comparison	Z	P.unadj	P.adj
1	Aleatorio - Ascendiente	-0.02465021	0.98033397	0.98033397
2	Aleatorio - Descendiente	1.79010648	0.07343680	0.08159644
3	Ascendiente - Descendiente	1.81475669	0.06956131	0.08695164

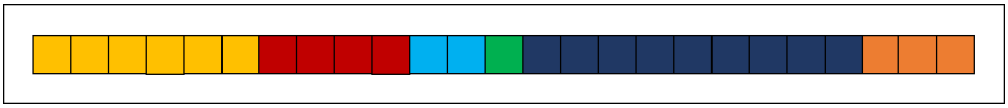
Esto tiene una relevancia muy grande pues no importa si las tareas estan ordenadas, es equivalente a cualquier ordenamiento (representado por el aleatorio). Recuerde que el caso nadir y el ideal no son ordenes de tareas sino pruebas con valores que esperamos le cueste menor o mayor trabajo que el caso general. Observe también como la variabilidad disminuye en el caso de utilizar 7 núcleos; en éste punto específico,sólo se distinguen dos grupos: caso ideal-nadir y los ordenes,los resultados no significativos de una prueba de Dunn para éstos datos aparecen enseguida:

```
Comparison                          Z          P.unadj      P.adj
1      Aleatorio - Ascendiente  0.23431554 0.8147400 1.0000000
2     Aleatorio - Descendiente  0.03791189 0.9697579 0.9697579
3  Ascendiente - Descendiente -0.19640365 0.8442942 1.0000000
10        MejorCaso - PeorCaso -0.10997138 0.9124321 1.0000000
```

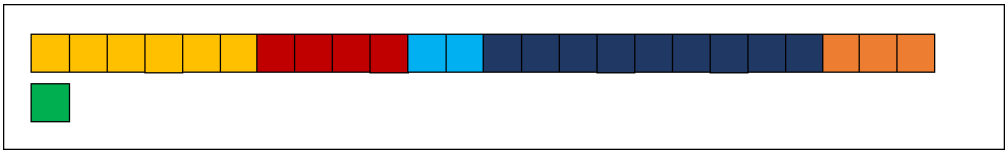
4. ALGUNAS INTUICIONES DEL PORQUÉ DE LOS RESULTADOS...

Sin embargo; aún persiste la pregunta: ¿porqué tener más núcleos para procesar no acelera la terminación del trabajo? Pues bien, una posible respuesta es la forma en que se asignan las tareas a los núcleos. Recuerde que no solo depende de como lo ordenemos sino de como los procese internamente R. Para ejemplificar hagamos la suposición de que las tareas se asignan una inmediatamente después de otra; es decir, estamos suponiendo que el núcleo del procesador termina de determinar si n_i es primo e inmediatamente se pone a determinar si lo es n_j. Estamos suponiendo que los núcleos únicamente se dedican a procesar nuestras tareas, cosa que por supuesto no es cierto. En fin, si no hay tiempo de preparación (como se conoce en *schedulling*), cuando usemos un solo núcleo pues no importa la secuencia en que se asignen las tareas, el tiempo de proceso deberá ser la suma de los tiempos de proceso de todas las tareas, como lo ilustra la Figura 4.1a

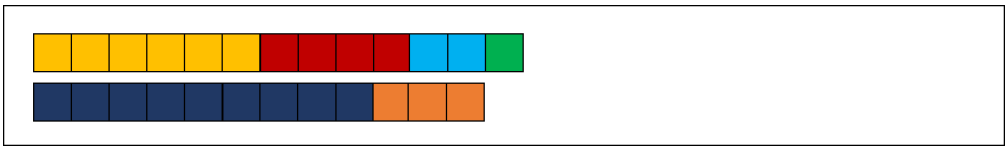
Es claro que al utilizar un núcleo más el tiempo de proceso debe disminuir siempre pues en el peor caso sólo la tarea más pequeña es asignada al segundo núcleo, como lo ilustra la Figura 4.1b. En promedio, la disminución de tiempo debe ser significativa en un caso como el de la Figura 4.1c. Sin embargo; a partir de tres núcleos la situación es muy distinta, el tiempo puede disminuir en un caso como el de la Figura 4.1d, donde sólo la tarea más pequeña se asigna al nuevo núcleo. Pero también puede aumentar el tiempo de proceso como se aprecia en la Figura 4.1e. Puede apreciarse que es fácil encontrar la forma de hacer aumentar el tiempo conforme se agendan las tareas.



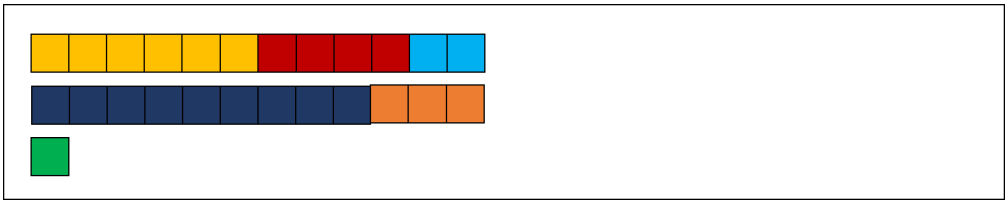
(a) Secuencia en un núcleo



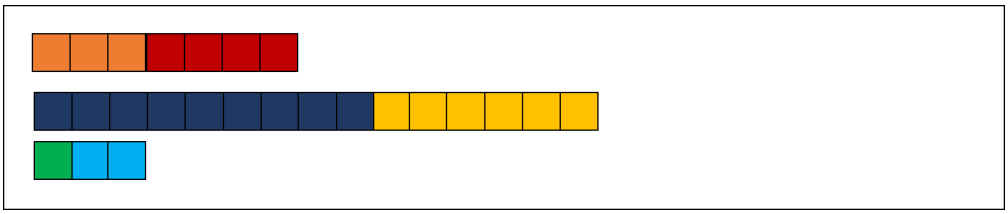
(b) Secuencia en dos núcleos peor caso



(c) Secuencia en dos núcleos caso promedio



(d) Secuencia en tres núcleos que mejora



(e) Secuencia en tres núcleos que empeora

Figura 4.1: Ejemplos de calendarizaciones para distintos núcleos disponibles