Relatório - Mini EP04

MAC0219

criado por Juliano Garcia de Oliveira, NUSP 9277086 (Maio de 2018)

Para testar as diferenças entre os algoritmos de seção crítica *Bakery* e *Gate*, utilizei um computador com as seguintes especificações:

- OS: Debian GNU/Linux 9.3 (stretch) x86_64
- CPU: Intel i5-4210U (x4) @ 2.7GHz
- RAM: 6 GB

Para coletar as medidas, modifiquei a impressão do código de forma a gerar um CSV, e então com os resultados obtidos, criei um script Python para gerar gráficos e calcular algumas estatísticas. Para escolher as estatísticas, rodei ambos os algoritmos no meu computador, e a quantidade de acessos à SC por cada Thread não variava muito nos 30 testes (em relação ao valor obtido nos 30 testes por cada Thread individualmente). Então, para observar a distribuição dos acessos à SC, é obtido a média de acessos à SC nos 30 experimentos por cada Thread, e com cada média, fiz o gráfico de barras para visualizar a distribuição entre as Threads.

Além desta estatística, para visualizar o quão "uniformemente"distribuídos está sendo o acesso à SC, é calculado o desvio padrão dessas médias. Quanto menor o desvio padrão, mais "uniforme"é a distribuição da *workload*. Importante notar que dependendo dos resultados, tirar o desvio padrão das médias não ajudaria, mas como nos experimentos que eu fiz cada thread possui uma quantidade de acesso não discrepante nos 30 testes, faz sentido observar o desvio padrão da média dos 30 testes. Por fim, também observando o tempo gasto por cada um dos 30 testes, vi que também não são muito diferentes entre si, então calculei o intervalo de confiança com 95% de precisão para os 30 testes. O tempo é expressado em microsegundos (10^{-6} segundos). Todos os resultados foram arredondados com 5 posições decimais de precisão.

O primeiro teste que eu fiz foi com número de threads entre 8 e 20, e um tempo na SC alto para tentar observar as diferenças. Abaixo, está um teste deste tipo:

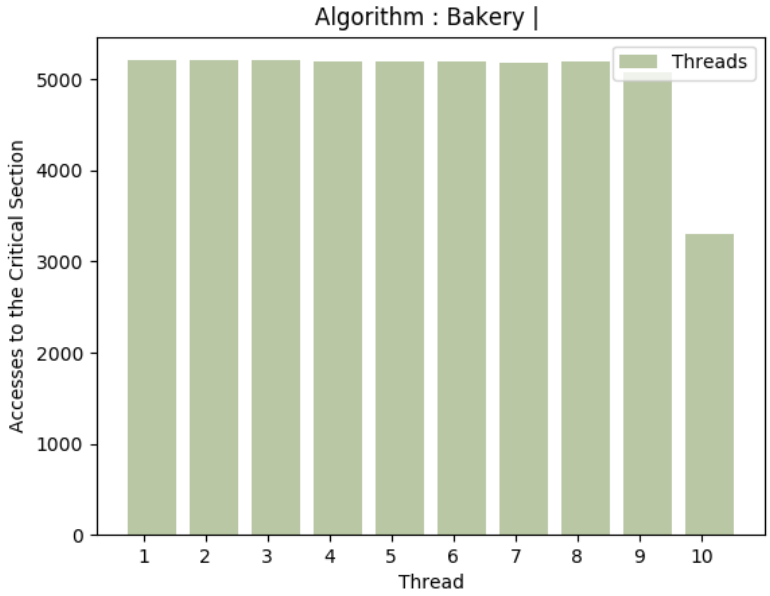
Execução: **./main 10 5000000**

Escalonador: **SCHED_OTHER**

Os resultados deste primeiro teste mostram que a distribuição que o algoritmo Bakery faz é muito mais 'uniforme' que o feito pelo 'Gate'. Basta olhar o desvio padrão das médias, onde fica claro que no caso do Gate, é bem mais alto, variando muito. Olhando o gráfico, fica claro que algumas poucas threads tem carga de trabalho bem maior que outras. O algoritmo Bakery, mesmo que distribua razoavelmente o *workload*, também não distribui perfeitamente, sendo possível ver que uma das threads trabalhou quase metade do que as outras. Há também uma diferença gritante entre o tempo gasto por ambos os algoritmos: O algoritmo Gate é quase 5 vezes mais rápido que o Bakery!

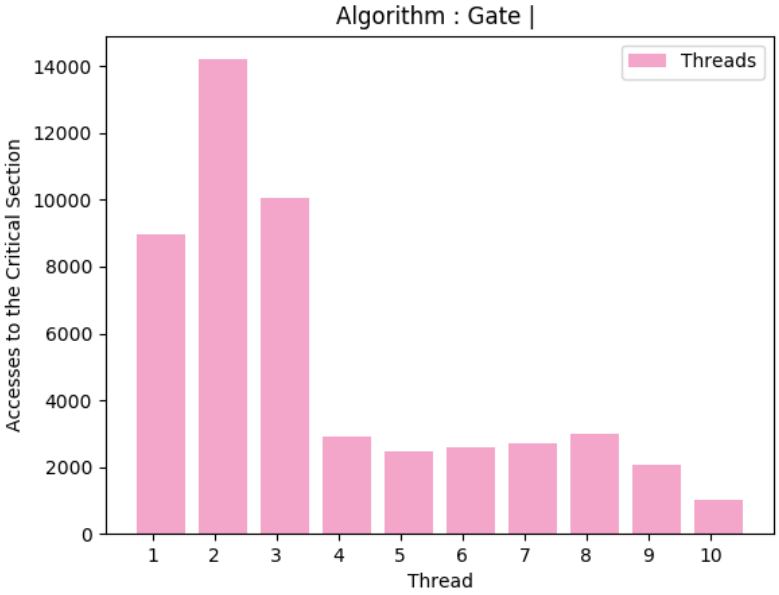
As imagens e números deste teste estão exibidas abaixo.

•**Bakery**



- Desvio padrão das médias: 567.76294
- Média das médias de acesso: 5000.9
- Intervalo de confiança do tempo, tempo médio: [114068.81565, 138615.88940] , 120454.93851 microsegundos

•**Gate**



- Desvio padrão das médias: 4196.57185
- Média das médias de acesso: 5000.82666
- Intervalo de confiança do tempo, tempo médio: [23691.41104, 25126.50288], 3070.96365 microsegundos

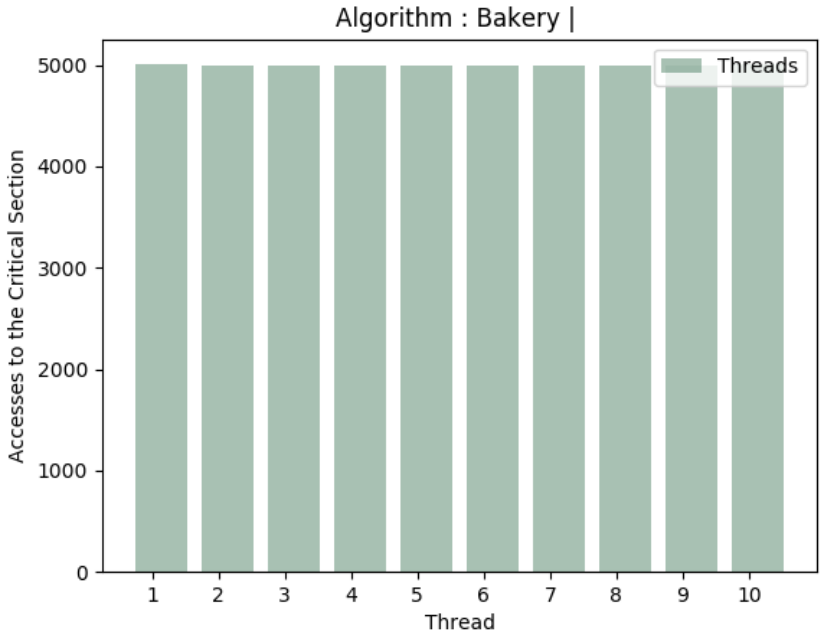
Refiz o mesmo teste acima, porém agora alterando o escalonador de SCHED_OTHER para SCHED_FIFO e SCHED_RR. Os resultados do FIFO ficaram bem parecidos com o Round Robin, então irei exibir apenas os do RR (o FIFO é similar, porém um pouco pior que os do RR).

Execução: **./main 10 5000000**

Escalonador: **SCHED_RR**

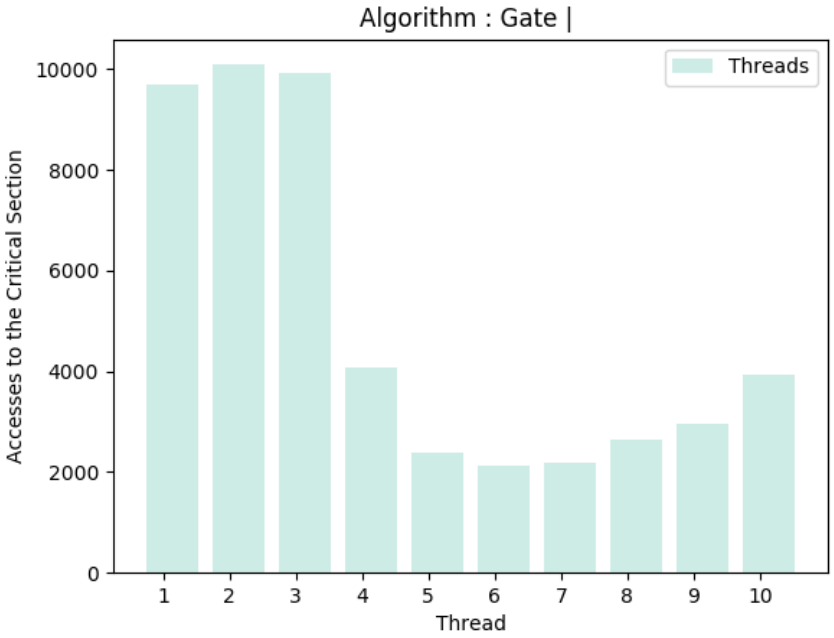
Neste segundo teste, percebe-se que o desvio padrão das médias diminuiu para ambos os testes, em relação ao SCHED_OTHER. Também vemos que o tempo para o Bakery diminuiu em relação ao teste anterior. No gráfico, percebe-se como a distribuição de *workload* é bem uniforme no caso do Bakery, e também uma pequena melhora no caso do Gate. Porém, o Gate continua distribuindo não uniformemente o trabalho entre as threads, onde observamos novamente que algumas poucas threads estão fazendo a maioria do trabalho.

•**Bakery**



- Desvio padrão das médias: 1.94861
- Média das médias de acesso: 5000.9
- Intervalo de confiança do tempo, tempo médio: [20234.17254, 25418.52438], 23431.93851 microsegundos

•**Gate**



- Desvio padrão das médias: 3268.88806
- Média das médias de acesso: 5000.9
- Intervalo de confiança do tempo, tempo médio: [2876.35491 , 3144.86262], 2925.58825 microsegundos

E para finalizar os testes, fui aumentando o número de threads e mudando o tempo. Com tempos pequenos, ocorria quase sempre a *starvation*, então não obtive resultados relevantes para observar a diferença entre ambos. Apenas aumentando o tempo, obtinha resultados parecidos com o primeiro teste, porém mudando as escalas. Abaixo está um teste com 500 threads, e usando o escalonador Round Robin, pois foi o mais 'justo' nos testes feitos.

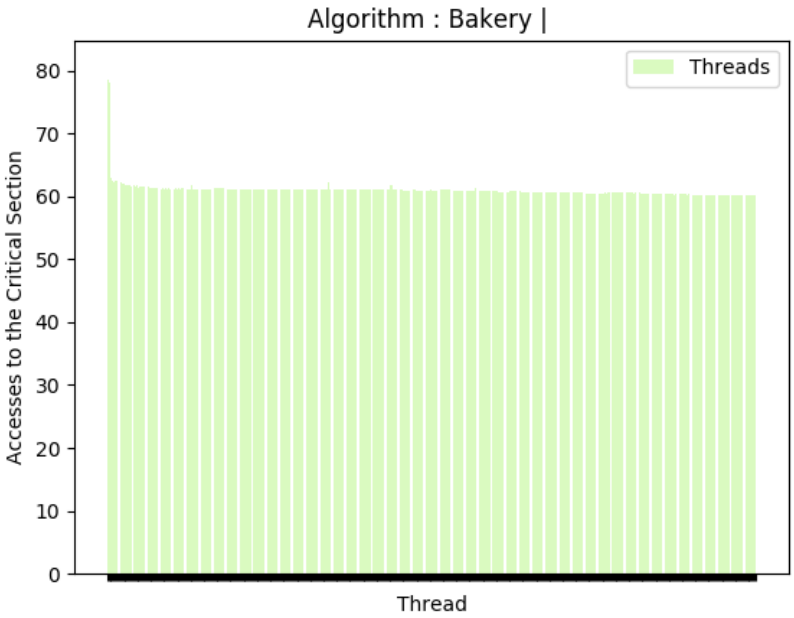
Execução: **./main 500 3000000**

Escalonador: **SCHED_RR**

Neste teste final, é possível ver ainda mais a disparidade entre os dois algoritmos. No Bakery, vemos que o desvio padrão é baixo, o gráfico mostra que há uma certa uniformidade na distribuição de *workload* (mesmo que as primeiras threads tenham um pouco mais de trabalho). Também vemos o tempo gasto pelo algoritmo, que se comparado com o Gate, é muito mais demorado.

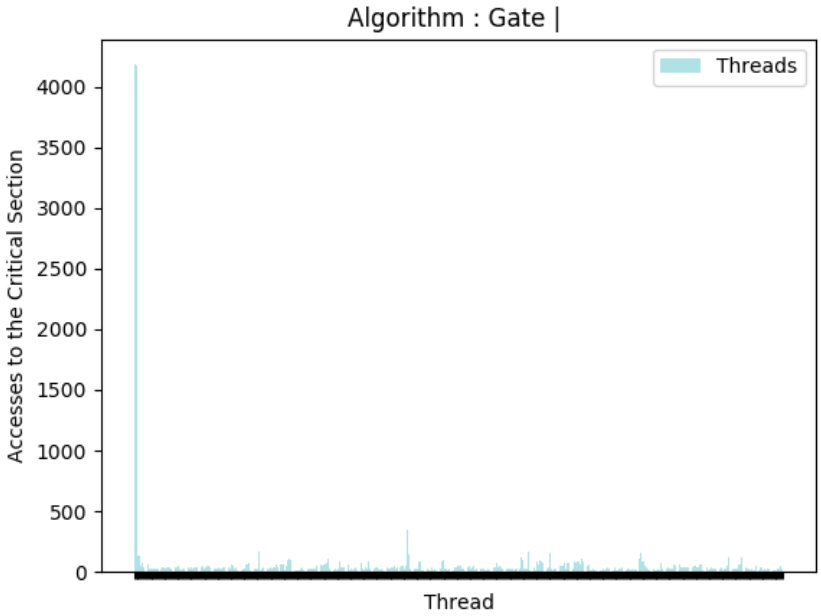
No algoritmo Gate, vemos claramente no gráfico como é o comportamento deste algoritmo conforme se aumenta as threads. Devido a não uniformidade da distribuição do *workload* entre as threads, fica até difícil ver no gráfico a quantidade de acessos das outras threads. O desvio padrão das médias, se comparado com o Bakery, é evidência disto. Porém, o tempo do Gate continua sendo bem mais rápido que o Bakery, neste caso chegando a ser aproximadamente 75 vezes mais rápido.

•**Bakery**



- Desvio padrão das médias: 1.80914
- Média das médias de acesso: 60.998
- Intervalo de confiança do tempo, tempo médio: [1055594.59546, 2181413.07467], 1482160.71403 microsegundos

•**Gate**



- Desvio padrão das médias: 283.29488
- Média das médias de acesso: 60.998

- Intervalo de confiança do tempo, tempo médio: [5677.49482, 20748.44447], 19610.48192 microsegundos

Em conclusão, primeiramente temos que, para ambos os algoritmos, utilizar o SCHED_RR é o que mais melhora a performance do algoritmo, e portanto, o mais 'justo' (como descrito no código *main.c*). Em segundo lugar, quanto maior o número de threads, além de ambos os algoritmos demorarem mais, a relação tempoBakery/tempoGate aumenta também, isto é, a proporção de tempo não se mantém constante. Exemplo disto é comparar o primeiro teste (10 threads) com o último (500 threads). Em terceiro lugar, em todos os casos de teste, o Bakery distribui a *workload* bem mais uniformemente do que o Gate, e a 'má distribuição' do Gate tende a piorar com o aumento das threads. O tempo gasto na SC influencia na medição, sendo que tempos pequenos não permitem fazer uma boa análise do que acontece com cada algoritmo, muitas vezes ocorrendo *starvation*. Porém, a partir de um certo tempo (por volta dos 3000000, como explicado no enunciado), aumentar mais esse número não muda muito as conclusões que tiraríamos.

Não coloquei os resultados ao se retirar a primitiva *__sync_synchronize()*, porque praticamente todas as vezes que retirava e rodava os testes um dos dois (Bakery ou Gate) não funcionava. Já foi explicado em sala que essa primitiva transforma instruções em "atômicas", criando o que é chamado de "barreira de memória", que força uma certa ordem na execução do código, o que é necessário nos algoritmos descritos. A main do programa em C calcula algumas estatísticas da medição, entre elas as que foram mais úteis para a realização da minha análise foram o tempo, média de acessos, e claro, a quantidade de acessos de cada thread à SC.