Trabalho 06

Disciplina: Programação Paralela **Data: 13 de maio de 2014.** **Prof.:** Fernando Castor

1. Implemente uma trava TAS com *backoff* exponencial, usando a interface `Lock` como base. Sua trava deve funcionar da maneira **mais eficiente possível** e não precisa ser reentrante. Crie um teste que usa essa implementação para proteger o acesso a dez objetos contadores que possuem apenas um método, `incrementa()`, auto-explicativo. Construa um programa onde, durante 2 minutos, 10 threads executam o método `incrementar()` repetidamente. Quantas vezes cada thread conseguiu executar `incrementar()`? Considere que a escolha do objeto a partir do qual cada thread fará o incremento é aleatória. E se forem 50 threads? E 100? E 200? Agora desligue a política de *backoff* que você utilizou. Como o número de execuções foi afetado em cada caso? E se sua política de *backoff* fosse aditiva, ao invés de exponencial? Agora retire o limite de tempo e faça com que cada thread execute o método `incrementar()` 1.000 vezes. Qual o tempo de execução em cada um dos cenários descritos anteriormente? Compare o desempenho da sua trava com a da classe `ReentrantLock` de Java. Apresente os resultados para todos esses casos e lembre-se que, para experimentos com desempenho, várias execuções são necessárias!

2. Agora implemente uma trava de fila tão eficiente quanto possível. E repita o experimento anterior utilizando-a. Os resultados mudaram? Por quê?

3. Exercício 86 do AMP.

3. Escolha uma entre as travas que você implementou para as questões 1 e 2. Modifique a implementação para que, no momento do travamento, seja verificado se a realização desse travamento resultaria em um deadlock entre uma ou duas threads acessando, respectivamente, uma ou duas travas. Se um deadlock for detectado, uma exceção deve ser lançada em pelo menos uma das threads envolvidas (idealmente em ambas, nos casos de deadlock com duas threads). Avalie experimentalmente o impacto dessa modificação no desempenho da operação de travamento quando deadlocks **não** ocorrem (ou seja, o *overhead* dessa modificação).

4. Analise experimentalmente o desempenho de sua implementação do *hashed deque* (aula anterior) quando varia-se o número de buckets e o uso de uma trava global vs. duas travas, uma para cada lado do deque.