Universidade Federal de Pernambuco
Centro de Informática
Graduação em Ciência da Computação

Um estudo sobre a incidência de bugs relacionados a deadlocks em aplicações C# de código aberto

Rafael Acevedo de Aguiar
raa7@cin.ufpe.br

Proposta de Trabalho de Graduação

Orientador: Fernando José Castor de Lima Filho

Recife
Abril de 2016

Resumo

Linguagens de programação de alto nível são o padrão de fato para desenvolvimento de sistemas de software[1], o que torna cada vez mais abstrata a interação do programador com a máquina. Isso torna possível que sistemas cada vez mais complexos sejam construídos. Para fazer maior uso do potencial de processadores multi-núcleo, processadores que são capazes de computar várias instruções ao mesmo tempo, são utilizadas threads para computar de forma paralela.

Enquanto computar de forma paralela traz um ganho significativo de desempenho[2], também traz mais dificuldades para quem está programando, visto que o programador deve se preocupar com o acesso destas threads aos recursos do sistema. Esta preocupação é real, pois caso a sincronização desse acesso esteja errada, pode levar a bugs como condições de corrida a deadlocks[3]. Se sincronização for sub-utilizada, duas ou mais threads podem acessar uma mesma região de memória ao mesmo tempo, o que pode levar a inconsitências. Por outro lado, excesso de sincronização pode causa problemas de desempenho[4]. Por fim, uma ordenação de operações de sincronização que torna possível que duas ou mais threads fiquem bloqueadas, cada uma esperando por um recurso que a outra não pode liberar, leva à ocorrência de um deadlock. Deadlocks podem ser extremamente difíceis de ser identificados. Por isso foram criadas varias abordagens de detecção e prevenção de deadlocks, como detecção estática[5,6] e dinâmica[7,8].

Uma das vantagens de projetos open-source é que toda a comunidade de desenvolvedores pode contribuir de alguma forma. Isto também se aplica a reportar bugs e discutir possíveis soluções. Como dito anteriormente, deadlocks são difíceis de detectar, assim, a ajuda de ferramentas e da comunidade é de extrema relevância. Um estudo que aborda isto é apresentado em [9], onde são apresentadas análises de bugs relacionados a deadlock em Java, além de uma implementação de travas[10] que trata deadlocks como exceções em tempo de execução. Este tratamento de deadlocks como exceções ajuda bastante na detecção da causa do deadlock, pois, em [9], a exceção levantada traz informações cruciais, como, por exemplo, as threads envolvidas no deadlock, facilitando a resolução de defeitos desse tipo.

Objetivos

O objetivo deste trabalho é analisar bugs relacionados a deadlocks em aplicações C# open-source. Assim, o foco do estudo é levantar informações relevantes para a comunidade, como: o tipo do deadlock existente, quais threads estão envolvidas, se o bug é realmente relacionado a um deadlock, e, caso possível, abordar maneiras para solucionar o defeito analisado.

Será realizado um estudo sobre quais serão as aplicações usadas para obtenção de defeitos que se relacionam com deadlock. Após esse levantamento,os bugs serão divididos em categorias, de maneira similar ao que é feito em [9], categorizando-os para facilitar a análise que seguirá.

Em seguida, serão analisados profundamente os bugs selecionados a fim de obter as informações desejadas pela comunidade: tipo do deadlock(recurso, comunicação, etc), as threads envolvidas no deadlock(isto pode ser feito manualmente ou com ferramentas, como o jstack[11]).

Cronograma

As atividades pretendem ser desenvolvidas conforme o cronograma abaixo.

Atividade	Março				Abril				Maio				Junho			
Revisão bibliográfica																
Levantamento aplicações para estudo																
Pesquisa e categorização dos bugs																
Análise dos bugs																
Análise de resultados obtidos																
Elaboração do relatório final																
Preparação da defesa																
Defesa																

Lista de possíveis avaliadores

Henrique Rebelo
Kiev Gama
Sérgio Soares

Assinaturas

_____

Fernando José Castor de Lima Filho
Orientador

_____

Rafael Acevedo de Aguiar
Aluno

Referências

[1] KIM, Larry. **10 Most Popular Programming Languages Today.** 2015. Disponível em:<http://www.inc.com/larry-kim/10-most-popular-programming-languages-today.html>. Acesso em: 14 abr. 2016.

[2] AGARWAL, A.. Performance tradeoffs in multithreaded processors. **Ieee Transactions On Parallel And Distributed Systems.** [s. L.], p. 525-539. set. 1992. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=159037&url=http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=159037>. Acesso em: 14 abr. 2016.

[3] DEADLOCK. Disponível em: <https://en.wikipedia.org/wiki/Deadlock>. Acesso em: 12 abr. 2016.

[4] Gu, Rui, et al. "**What change history tells us about thread synchronization.**"Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. ACM, 2015. <http://dl.acm.org/citation.cfm?id=2786815> Acesso em: 14 abr. 2016

[5] Williams, Amy, William Thies, and Michael D. Ernst. "**Static deadlock detection for Java libraries**." ECOOP 2005-Object-Oriented Programming. Springer Berlin Heidelberg, 2005. 602-629. <http://people.csail.mit.edu/amy/papers/deadlock-ecoop05.pdf> Acesso em: 14 abr. 2016

[6] Naik, Mayur, et al. "**Effective static deadlock detection.**" Proceedings of the 31st International Conference on Software Engineering. IEEE Computer Society, 2009. <http://www.cc.gatech.edu/~naik/pubs/icse09.pdf> Acesso em: 14 abr. 2016

[7] Li, Tong, et al. "**Pulse: A Dynamic Deadlock Detection Mechanism Using Speculative Execution.**" USENIX Annual Technical Conference, General Track. Vol. 44. 2005. <http://static.usenix.org/event/usenix05/tech/general/full_papers/li/li_html/> Acesso em: 14 abr. 2016

[8] Joshi, Pallavi, et al. "**A randomized dynamic program analysis technique for detecting real deadlocks.**" ACM Sigplan Notices 44.6 (2009): 110-120. <http://www.cc.gatech.edu/~naik/pubs/pldi09b.pdf> Acesso em: 14 abr. 2016

[9] Lobo, Rafael, and Fernando Castor. "**Deadlocks as Runtime Exceptions.**"Programming Languages. Springer International Publishing, 2015. 96-111. <http://link.springer.com/chapter/10.1007/978-3-319-24012-1_8> Acessado em: 13/04/2016

[10] LOCK. Disponível em: <https://en.wikipedia.org/wiki/Lock_(computer_science)>. Acesso em: 14 abr. 2016.

[11] ORACLE. **Jstack.** Disponível em: <https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstack.html>. Acesso em: 14 abr. 2016.