6
Conclusão

O desenvolvimento e a manutenção de sistemas distribuídos são tarefas complexas, em razão do não determinismo associado à execução deste tipo de aplicação, que está sujeita a erros provenientes de condições de corrida, da latência de comunicação e de falhas parciais. A ferramenta proposta nesse trabalho é apenas uma parte de um conjunto de ferramentas que o desenvolvedor deve ter a sua disposição, a fim de que ele possa entender o comportamento do sistema, especialmente para apoiar uma manutenção ou correção de um problema.

A ferramenta desenvolvida neste trabalho faz a reconstrução das sequências de interações entre os componentes e permite a análise de vários aspectos do sistema. Essa análise pode ser feita dinamicamente (*on the fly*) ou *post morten*, pois as informações coletadas durante a execução do sistema são processadas, organizadas e armazenadas, permitindo assim, consultas futuras. Essa ferramenta permite visualizar as sequências de chamadas remotas decorrentes das interações entre os diversos componentes da aplicação, associadas a informações sobre a topologia da aplicação. Permite, ainda, a análise de informações sobre o desempenho e outros aspectos da aplicação a partir de consultas feitas a uma base de dados.

Na ocorrência de um erro ou de um comportamento inesperado da aplicação, o desenvolvedor poderia utilizar a ferramenta proposta para analisar as informações disponíveis, a fim de identificar a origem e a causa do problema, restringindo pouco a pouco o escopo de sua busca.

Considerando o nosso estudo de caso, a ferramenta se mostrou capaz de apresentar respostas para quase todas as questões levantadas juntos aos desenvolvedores relativas à execução da aplicação. A resposta a essas questões poderia auxiliar o desenvolvedor a raciocinar sobre um dado problema e entender o comportamento do sistema em execução, o que permitiria, ao desenvolvedor, restringir o conjunto de componentes que podem ter contribuído para a ocorrência do problema observado.

Uma limitação deste trabalho se deve ao fato de que a ferramenta proposta não foi avaliada em um cenário de falha de uma aplicação. Falta um

experimento na qual seja injetada uma falha na aplicação para verificar se as informações disponíveis podem realmente auxiliar o desenvolvedor a reduzir o conjunto de possíveis componentes causadores da falha. Os experimentos realizados apontam em um direção positiva mas falta um experimento conclusivo nesse sentido específico.

Por fim, devemos fazer algumas considerações. Em primeiro lugar, a instrumentação da aplicação causa uma inevitável redução no desempenho. Seu uso está condicionado a uma avaliação do impacto da sobrecarga imposta pela instrumentação, a fim de que seja decidido se essa sobrecarga pode ser tolerada ou não pela aplicação. Essa análise depende da aplicação a ser instrumentada. Em segundo lugar, o componente *TraceLogger* pode se tornar um gargalo no caso de aplicações maiores, o que poderia comprometer a escalabilidade da ferramenta desenvolvida neste trabalho.

6.1 Trabalhos Futuros

Nesse trabalho não exploramos a interface gráfica da ferramenta. O visualizador desenvolvido é um exemplo de como os eventos gerados pelo componente *TraceLogger* podem ser usados para facilitar a análise do sistema por parte do desenvolvedor. O projeto de uma interface gráfica que seja mais intuitiva para o desenvolvedor e com mais recursos seria uma extensão natural deste trabalho. A interface poderia oferecer filtros para as informações, permitindo que o usuário pudesse focar em transações que envolvem um dado componente, ou conjunto de componentes, ou ainda filtrar ou agrupar as transações por períodos de tempo.

Também seria útil que a ferramenta permitisse aumentar ou diminuir a granularidade da análise, fazendo por exemplo, com que as transações relativas a componentes de um mesmo serviço ou pertencentes a um mesmo *container* (no caso de uso da infra-estrutura de execução) pudessem ser agrupados.

Outro melhoramento seria oferecer a possibilidade do usuário atribuir uma semântica para os métodos, associando-os a um evento específico da aplicação. Considerando o nosso estudo de caso, o usuário poderia associar ao método `loginByPassword` do componente *AccessControlService* um aviso que o serviço que realizou a chamada remota conectou-se ao barramento, no caso da chamada ter sido bem sucedida, ou que tentou a conexão, mas esta foi recusada, caso a chamada retorne uma exceção. Dessa forma a ferramenta notificaria ao usuário eventos que carregam uma maior carga semântica.

Outro exemplo similar seria associar ao método `renewLease` o evento do tipo *"ainda vivo"*, ou seja, cada vez que o componente chamasse esse método

para renovar sua credencial, a ferramenta saberia que o componente ainda está conectado ao barramento. Caso algum componente deixasse de executar aquele método por um período de tempo maior do que aquele definido para que a credencial expire, a ferramenta poderia notificar ao usuário que aquele componente não tem mais uma credencial válida e, portanto, perdeu acesso ao barramento.

Seria possível, ainda, a implementação de uma ferramenta que utilizasse dinamicamente as informações geradas pelo componente *TraceLogger* para indicar ao desenvolvedor um conjunto de possíveis causas do erro e sugerir medidas para solucioná-lo. Nessa linha, podemos destacar o trabalho de Mariani e Pastori [24] que propõem uma técnica para análise automática dos arquivos de *logs* com o objetivo de identificar informações relevantes para identificação das causas de uma falha, utilizando para isso um modelo baseado em autômatos finitos, que representa a sequência de eventos.

Dentro do estudo de caso utilizado neste trabalho, as questões levantadas pelos desenvolvedores foram mapeadas em uma consulta SQL a ser feita a base de dados. Outra possível extensão deste trabalho seria a utilização de uma interface mais intuitiva para a criação de consultas, utilizando uma linguagem de consulta mais próxima do domínio da aplicação analisada.

Como discutimos na seção 5.2, uma limitação de nossa ferramenta é a ausencia de associação das transações com os eventos específicos da aplicação. Outro trabalho futuro seria suprir essa limitação para permitir a correlação entre uma função remota invocada e o *log* da aplicação.

É ainda necessário estudar formas de evitar que o *TraceLogger* se torne um gargalo, comprometendo a escalabilidade de nossa ferramenta. Nesse sentido, uma possível abordagem seria a de balancear a carga entre diversas instâncias desse componente.

Por fim, existe ainda espaço para alguma otimização do código presente nos pontos de interceptação. Como esse código é executado a cada chamada remota qualquer ganho de desempenho pode ser relevante, principalmente no caso de aplicações que fazem muitas chamadas remotas, com pouco processamento.