**Main contributions:**

**1. The authors would need to identify the key contribution: whether it's the dependency graph identification or the replay performance improvement, and better focus on such identified contribution:**

1. **PaaS (mainly web applications with database):** Tem de se perceber que é para sistemas distribuidos e que é dai que vêm as vantagens.

**Dependency Tracking:**

**1. While it's claimed that Shuttle (the technique) is for PaaS, still the technique relies on logs of HTTP requests and snapshots of database. Those are obviously application-specific especially when it comes to identify the dependencies between the requests and the updated database records/columns. At least, the dependency graph subsection's not well explained in an easy way. Some algorithms and examples (that clearly show how the presented techniques transparently track the dependencies) would be necessary.**

Claramente não percebeu a secção das dependencias…

**Related Work:**

Fez faltam a secção de background e de related work…

**1. Qual a diferença para o [17, SOSP2011 - Intrusion recovery for database-backed web applications]? The differences are stated well in the paper but may not with sufficient supporting data.**

**2. It'd be also helpful if more detailed explanation and comparison of [12-19] is provided in a dedicated related work section.**

**3. No related work: I am simply not able to get a holistic view of what has been done in the past and how the authors improve the current situation. Could we just combine the previous works, adapted to this particular problem with a few simple changes, and get the solution?**

**4. The paper could do a better job of discussing and comparing with various (a long list) recovery techniques. Very little exists in that regard. Two things were not clear. How and when is an intrusion detected/determined? Is it similar or a long running transaction? Second, what exactly does a state of the system consist of?**

**Gerais:**

**1) There are grammatical issues in the abstract and introduction.**

Estarão resolvidas nesta nova versao?

**2. Security/privacy: Some requests may have sensitive data that should not be stored on the servers, for example credit card data in some countries. Taking the example of credit cards, you would normally just complete the transaction and then forget the credit card number. Now, you would be storing it.**

Extremamente válido. É um problema de confidencialidade. Honestamente, não sei como argumentar contra. Dizer que os pedidos não são registados? Ter 2 serviços? Um para pagar o outro para o resto? Acho que não justifica aprofundar isto no paper…

**4. Incompleteness: Application state may not just be in databases. Even files on the filesystem also need to be covered. At least some discussion is needed. . (Tese: check)**

Assumimos apenas em base de dados. O conceito de aplicação para PaaS é usar bases de dados e os seviços de storage blob (que são em grande parte semelhantes a bases de dados)

*We assume applications to store their persistent state only in databases. Shuttle's architecture can be extended to encompass object storage, for instance \acf{aws} \ac{s3}. We do not consider a possible state stored in the filesystem because \ac{PaaS} applications are supposed to be scalable, thus the instances file system is frequently destroyed.*

**5. What happens if there are intrusive attacks during the replay process? Couldn't understand. (Tese: check)**

O processo tem de ser reiniciado numa nova branch e os pedidos de intrusão não podem ser re-executados. Em alterantiva, no inicio do restrain, os pedidos maliciosos já têm de estar identificados e não são re-executados.

**6. Single point of failure: Are we making a decentralized application more vulnerable to failures because of requirements of single proxies and load balancers? (Tese: check)**

Argumento válido, correcto e esperado… Ter o single-proxy tem esse problema. 2 proxies sincronizados funcionam bem mas não é referido…

The decentralized applications are more vulnerable to failures because of the single proxy architecture. However, we argue that future architectures can consider replication of the proxy, load balancer, Shuttle Storage and database.

**Evaluation:**

**1. O caso do XSS attack [17, SOSP2011 - Intrusion recovery for database-backed web applications], como é que é resolvido?**

O caso do XSS é escolhido para o [17] porque o [17] faz replay no browser do client: se o código injectado não existir, a re-execução é diferente porque o replay é feito no browser. Para o Shuttle é possível mas os pedidos maliciosos têm de ser identificados e corrigidos (uma brainstorm por skype é mais simples)

**2. Accuracy evaluation needs a better metric (e.g., ratio) that shows how much of the legitimated requests are restored and how much of the changes made by identified malicious requests were rolled back. 3) If you think about Table I is not really showing accuracy. It is showing the benefits of the selective replay algorithm. Might consider changing the caption.**

Mudar a caption e mesmo a própria tabela…

**3. Another evaluations are thorough though the baseline is not an existing work. It'd be great if the authors compare the performance of this system with [17] or other recent works.**

Re-escrever a accuracy para incluir os casos de [17]? Ou discutir apenas (tal como sugerido na pergunta 1.)?

**4. Requests have a size. the final thing the application stores may just be something small, but requests themselves could be big. For example, in a Dropbox-like application with file versioning, a request may contain a whole large file to be sync'ed. However, the application may save only a small chunk by applying delta compression on the previous version of the file that was already saved. (Tese-check)**

É um problema abordado na tese: a compress dos pedidos é enorme porque os pedidos são semelhantes. O sistema não está desenhado para aplicações que têm pedidos grandes. Pode ser adaptado, mas não está feito para isso. Está na secção 5.3.3 da tese.

**5. Lack of insights on real-world applications. The authors could have done better on this by evaluating a few real-world applications rather than developing their own. I suspect a Q/A app is on the simpler side. Can we consider other applications, such as office productivity, diagramming apps (these will probably have many user requests), and ecommerce apps including shopping cart apps. I do not know if there are such applications using distributed databases that can be tested but such experiments will be good. B) The paper could be improved by presenting more of intrusion scenarios and their implications. (Tese: check)**

Na tese deixei isso como future-work…. Quase que dá para fazer um paper só a falar sobre isso: as dependencias geradas dependendo do tipo de aplicações…

*We consider real-world applications such as office productivity, gaming, media content, data aggregation, wikis, diagramming and e-commerce applications (with shopping cart). For sake of simplicity, we developed \ac{Ask} instead of using real-world applications. Even \ac{Ask} implementation is based on the real-world application StackExchange, its implementation is simpler than it. This allows to analyze the dependencies and check the results. We expect to evaluate the dependencies created by different types of application in future.*

**6. Performance evaluation could be better, looking at real applications and their characteristics.**

Sim, os servidores aplicacionais são lentos e o serial replay também. Mas isto são protótipos.

**7. The presentation of workload is not clear, nor is it clear if what it is evaluated upon is very useful.**

Está mais clara na tese… Há pouco espaço ☹

**8. Assumptions about applications: Shuttle makes many assumptions such as those regarding non-determinism. Can we evaluate if such assumptions generally apply to most real-world applications? It may be difficult or not feasible but some discussion is warranted.**

Será que ele está a falar do não determinismo dos timestamps, etc? Ou ao nível do Replay?