**Questions about Distributed Systems:**

**How to test a distributed system?**

[**https://www.quora.com/How-do-I-test-a-distributed-system**](https://www.quora.com/How-do-I-test-a-distributed-system)

**In which case would you apply asynchronously communication between two systems?**

https://stackify.com/when-to-use-asynchronous-programming/

**What are the general pitfalls of Remote Procedure Call?**

*14*

*As a general rule, RPC provides a higher level of abstraction than some other means of interprocess communication. This makes it, perhaps, easier to use than lower level primitives. For this abstraction you may pay some penalty in performance due to marshaling/unmarshaling and may have to deal with added complexity in configuration for simple scenarios.*

*You might be interested in this thesis (pdf) by Jackie Silcock which discusses differences between message passing, RPC, and distributed shared memory with respect to several different measures of performance and implementation. You can also read one of the papers based on the thesis: Message Passing, Remote Procedure Calls and Distributed Shared Memory as Communication Paradigms for Distributed Systems (pdf)*

**If you are building a distributed system for scalability and robustness, what are the different things you'd think of in the case you are working in a closed and secure network environment or in geographically distributed and public system?**

**How to manage Fault Tolerance in a Web application? And in a Desktop one?**

**How to deal with failures in Distributed Systems?**

https://www.alibabacloud.com/blog/deploy-web-apps-with-high-availability-fault-tolerance-and-load-balancing-on-alibaba-cloud\_277149

**Let's talk about the several approaches to Reconciliation after network partitions**

**What are the Fallacies of Distributed Computing?**

[**https://www.quora.com/How-do-I-solve-classes-of-failures-in-distributed-systems**](https://www.quora.com/How-do-I-solve-classes-of-failures-in-distributed-systems)

**When would you use Request/Reply and when Publish/Subscribe?**

*Solution B only works if you request the data from System A every time it is required by System B. Otherwise you cannot guarantee the information is up to date. This is not as bad as it sounds and is in fact the basis of all SOA architectures.*

*The decision of whether to push (Solution A) or pull (Solution B) depends on the ratio of updates in system A to the number of queries in System B.*

*Where you have lots of updates in system A versus a few queries in System B then "pull" is the obvious choice.*

*Where you have relatively few updates in system A and lots of queries in System B then "push" is the way to go.*

*Where you have lots of updates and lots of queries either choice is equally good (or bad!).*