INNOPOLIS UNIVERSITY

# Distributed Systems Reading Questions 1

*Timur Samigullin*

supervised by
Konstantin Urysov

October 21, 2016

1. An alternative definition for a distributed system is that of a collection of independent computers providing the view of being a single system, that is, it is completely hidden from users that there even exist multiple computers. Give an example where this view would come in very handy.

The first example is DNS. When the client try to resolve an IP address, it ask only one DNS server. For end user this process is transparent, because, the same server are provide result to client, and client doesn't know about many requests to TLD servers, recursion and other things.

The second example is Google Global Cache (GGC). It enables company to optimize network infrastructure costs associated with delivering Google and YouTube content to companies users by serving this content from inside your network. The content delivering not from Google servers, that locates somewhere in USA, but from the local ISP, but client think, that is it delivered from Google.

The last example is anycast technology. The same ip address can be announced through different autonomous system. And it can be accessible by and users, which are near it.

2. What is the role of middleware in a distributed system?

Middleware for distribution system is like an operating system for single computer. It is a manager of resources offering its application efficiency share and deploy those resources across a network. It offers services, that can also be found in most operating systems, including: facilities for interapplication communication, security services, accounting services, masking of and recovery from failures. The most different, that middleware services are offered in network environment.

3. Explain what is meant by (distribution) transparency, and give examples of different types of transparency.

**Distribution transparency** is one of the most important property of distribution system, when a fact that its processes and resources are physucally distributed across multiple computers in the world. So distribution of processes and resources are transparent, i. e. invisible to end users and applications. There several types of distribution transparency: access, location, relocation, migration, replication, concurrency, failure.

Differences in naming conventions, differences in file operations, or differences in how low-level communication with other processes is to take place – are examples of **access** issues that should preferably be hidden from users and applications.

The example of **location** transparency is the uniform resource locator (URL), which gives no clue about the actual location of webserver. The example of **relocation** transparency is a communication between mobile phones: whether two people are actually moving, mobile phones will allow them to continue their conversation. The example of **concurrency** transparency is when to independent users may each have stored their files in the same file server, or when different applications accessing the same table in database. The example of **failure** transparency is when browser contacts with a busy web server, a browser will evaluate time out report. In this case, user cannot conclude, is it is a really problem with webserver, or just a network issue.

4. Why is it sometimes so hard to hide the occurrence of and recovery from failures in a distributed system?

It is generally impossible to detect whether a server is actually down, or that it is simply slow in responding. Consequently, a system may have to report that a service is not available, although, in fact, the server is just slow.

5. Why is it not always a good idea to aim at implementing the highest degree of transparency possible?

As distributed systems are expanding to devices that people carry around and where the very notion of location and context awareness is becoming increasingly important, it may be best to actually expose distribution rather than trying to hide it. An obvious example is making use of location-based services, which can often be found on mobile phones. There is also other argument against fully distribution transparency. It is not good when developers of applications, that work on a distributed system assume that it is not distributed. If they will not the behaviour of distributed system, it helps them to develop applications more clearly.

6. What is an open distributed system and what benefits does openness provide? To be open means that components should adhere to standard rules that describe the syntax and semantics of what those components have to offer. A general approach is to define services through interfaces using an Interface Definition Language (IDL). Interface definitions written in an IDL nearly always capture only the syntax of services. In other words, they specify precisely the names of the functions that are available together with types of the parameters, return values, possible exceptions that can be raised, and so on. The hard part is specifying precisely what those services do, that is, the semantics of interfaces. In practice, such specifications are given in an informal way by means of natural language. Another important goal of an open distributed system is that is should be easy to configure the system out of differet components, possible from different developers. Also it should

be easy to add new components, that stay in place. In other words, an distributed system should also be extensible.

7. Describe precisely what is meant by a scalable system. Scalable system is when current system can be increased with it functionals, that provide almost the same function for users. Scalability of a system can be measured along at least three different dimensions:

- Size scalability: A system can be scalable with respect to its size, meaning that we can easily add more users and resources to the system without any loss of performance.
- Geographical scalability: A geographical scalable system is one in which the users and resources may lie far apart, but the fact that communication delays may be significant is hardy noticed.
- Administrative scalability: An administrative scalable system is one that can still be easily managed even if it spans many independent administrative organizations.

8. Scalability can be achieved by applying different techniques. What are suitable techniques for size and geographical scaling? And for administrative scaling?

In most cases scalability problems appears as performance problem caused by limited capacity of servers and networks. To decrease negative effect of implementing geographical scalability, we need to decrease the distance between end users and applications, that provide services. For this reason we can conclude, that hiding communication latencies, replication and caching techniques can be used.

- hiding communication latency: trying to avoid waiting for response to remote-services requests as much as possible, to use asynchronous communications.
- replication: for geographically scalable distributed system it is a good idea to have some servers that provides services near endusers. It can be achieved by replication between different services.
- caching: it is the same idea as replication. We can locate some cache frontmost data near endusers to provide low latency.

Partition and distribution is suitable technique for size scalability. By subdivided the whole system info different parts, we can increase it performance, and avoiding single server to deal will all requests from endusers.

For administrative scaling, we should subdivide the whole system, or add new components geographically in places, where it can be administrate by experience staff, in other hands, the cost of service of this components should be less than in other region. It is just an example of administrative scaling.

9. We argued that distribution transparency may not be adequate for pervasive systems. This statement is not true for all types of transparencies. Give an example.

Thinking of migration transparency. In many pervasive systems, components are mobile and will need to re-establish connections when moving from one access point to another. Preferably, such handovers should be completely transparent. Likewise, it can be argued that many other types of transparencies should be supported as well. However, what should not be hidden is a user is possibly accessing resources that are directly coupled to the user's current environment.