4.2.9 Scalability

Return to Non-Functional Requirements

- ☐ **[char]Please Review**

About

Return to Top

Scalability is the ability of a system to accomplish more work while maintaining the quality (i.e., without degradation) of the products produced or services provided by the system. There are different ways to calculate the work produced or performed by the system and it usually depends on the kind of product produced or services rendered.

For software systems, here are some of the common metrics used to quantify the products produced or the services provided :

- Number of transactions or events per unit of time (i.e., 5,000 credit card transactions a second[1)])
- Number of tweets processed per unit of time (i.e., 6,000 tweets a second[2)])
- Number of hours of videos uploaded per minute (i.e., 500 hours of fresh video per minute[3)])
- Number of searches per day (i.e., 3.5 billion searches per day[4)]).

Scalability is about being able to increase the output of products and services without major disruptions, interruptions or increased costs. Often, because of the Economies of Scale, the estimates for the costs should actually decline.

There are different approaches to Scalability:

- Scaling Up is is generally applied to centralized or decentralized systems or products. It amounts to just adding more resources with more powerful versions. This works until you have reached the limits on the availability of more power. For example, the speed of the Central Processing Unit (CPU), memory or even the network. In decentralized (i.e., Cloud Computing) the scaling can either be Vertical or Horizontal Scaling.
- Scaling Out is usually associated with Distributed System which by its nature allows for more Network Nodes replicated with prepacked applications (i.e., Distributed Application (ĐApp or DApp)) to be added with minimal cost overhead. In essence, more nodes offering redundant products and services. Another approach is to divide up the problem by functionality. For example, if accessing customer data is the bottleneck, then add more nodes that access the same data. If the access to the actual data is an bottleneck, then adding replications to the data store is recommended.

Both Scaling up and scaling out are valid alternatives.

DDS Specifics

Return to Top

data_distribution_service_dds is a distributed Publish-Subscribe, Peer-to-Peer (P2P), Message-Oriented Middleware (MOM). The publishers and subscribers each join a DDS Domain and immediately start sending a receiving information without the need for setup or configuration. The mere act of publishing and subscribing is all that is needed for configuration. Each ddsnode is responsible for providing the computing resources necessary accomplish their tasks. These are generally small in terms of CPU, Memory, and disk space. As a consequence, to scale a solution, just add more nodes to either publish or subscribe. The only limiting factor is the network capacity. Remember, this is P2P, therefore the network capacity is directly related to the the amount of traffic that needs to flow over the network. Need more capacity, add extra Ethernet capacity.

In a server based MOM, there is a minimum of two messages for every message sent. One message between the publisher and the server, and another between the server and the publisher. In essence, this halves the network capacity.

DDS also uses an efficient binary transport mechanism (see DDS Interoperability Wire Protocol (DDSI-RTPS)). This further reduces the message traffic to the smallest size possible per message and almost eliminates the cost of marshalling and unmarshalling the data at either Endpoint. eXtensible Markup Language (XML) and JavaScript Object Notation (JSON) based messages are sent as American Standard for Information Interchange (ASCII) text that also include the names of each of the fields and delimiters further decreasing the efficiency over the wire.

dds_extensible_types_dds-xtypes provides a mechanism to add / remove /modify the Data Model (DM) of the each sensornode at runtime without the need for reconfiguring the entire network and the communications start in the middleware which automatically collects the configuration data of the configuration data from all the nodes creating a snapshot database from the latest live data from the network. The data can be accessed from the user application using a subscriber using a standard Application Programming Interface (API) Data Distribution Service (DDS). This allows the user application to remain agnostic to any node replacement in the network and the scaling of the network by updating the user application and processing the data without any coding of the application. And the sensor nodes can join or leave the network at any point in time. This means the nodes can dynamically discover that an update is needed without the need for Domain Name System (DNS).

DDS makes it easier for adding a node and deploying distributed application as the communication stack automatically collects data from all publishing nodes and creates a snapshot database of the latest live data from the network, which can be accessed from user application using standard APIs

1)

Ryan Vlastelica, Why bitcoin won’t displace Visa or Mastercard soon Market Watch, 18 December 2017, Accessed 10 August 2020, https://www.marketwatch.com/story/why-bitcoin-wont-displace-visa-or-mastercard-soon-2017-12-15

2)

Kit Smith, 60 Incredible and Interesting Twitter Stats and Statistics, Bandwidth, 2 January 2020,

Accesssed 10 August 2020, https://www.brandwatch.com/blog/twitter-stats-and-statistics/
3)

James Hale, More Than 500 Hours Of Content Are Now Being Uploaded To YouTube Every Minute, Tubefilter, 7 May 2019, Acessed 10 August 2020, https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/
4)

Maryam Mohsin, 10 Google Search Statistics You Need to Know in 2020 [Infographic], Oberlo, 3 April 2020, Accessed 10 August 2020, https://www.oberlo.com/blog/google-search-statistics

From:
https://omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:16_scalability&rev=1605579229

Last update: **2020/11/16 21:13**