Department of Computer Science
Computer Networks Research Group

Investigation of MANO scalability challenges

Management of ServiCes Across MultipLE clouds

Authors:

Ashwin Prasad Shivarpatna Venkatesh
Bhargavi Mohan
Deeksha Mysore Ramesh

Supervisors:

Prof. Dr. Holger Karl | Sevil Dräxler | Hadi Razzaghi Kouchaksaraei

Paderborn, April 17, 2019

Contents

List of Figures

1 Introduction

Scalability in the recent times has become one of the most important factors of the cloud environments. This document explains what scalability is and also gives a few insights about the effects of scaling a system and investigates some scaling approaches that could be incorporated to scale a MANO so as to implement MANO scalablity.

1.1 Definition of scaling

'Scalability' can be defined in different ways:

- It can be defined as [fur]"the ability of a particular system to fit a problem as the scope of that problem increases (number of elements or objects, growing volumes of work and/or being susceptible to enlargement)."
- Also can be defined as [LK] "Scalability of service is a desirable property of a service which provides an ability to handle growing amounts of service loads without suffering significant degradation in relevant quality attributes. The scalability enhanced by scalability assuring schemes such as adding various resources should be proportional to the cost to apply the schemes."
- Another definition states that [CMK] "Scalability is the ability of an application to be scaled up to meet demand through replication and distribution of requests across a pool or farm of servers."
- According to [noa] "A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity"

1.2 Why does a MANO need scaling?

In recent years, distributed systems have gained an increase in the number of users and resources. Scaling such a system is an important aspect when large user requests have to be served without compromising system performance or increase in administrative complexity. In terms of MANO, when there are a large number Network Service(NS) instantiations of various network functions, they need to be instatiated considering all the metrics to evaluate an ideal system. The metrics for scalaing are discussed in the section

1.3 System load

In a distributed system, the system load is the large amount of data that are to be managed by network services increasing the total number of requests for service. The load on a MANO can be defined in terms of it's load on NFV Orchestrator to process large number of tasks like on-boarding VNFs. The NFVO of a MANO also receives monitoring information which inturn increases the load on NFVO triggering it to scale the network service across multiple MANOs in a distributed system [STCP17].

1.4 Metrics to assess a scalable system

The definitions of scalability are already discussed earlier in this section. In this section, a few metrics in terms of a MANO server are introduced

- **Response time**: Service response time of a MANO is a time period from when a service invocation message is arrived to a MANO on the provider side to when a response for the invocation is returned to the service consumer.
- **Throughput**: It is a metric which measures the efficiency of a MANO to handle service invocations within a given time.
- **Cost**: High scalability under high service loads is an expensive affair. There is always some additional cost involved in planning a scalbility strategy.
- **Performance**: MANO should be able to handle the growing amount of service loads. Scalability should take into account MANOs' ability to manage high service loads without deteriorating Qos.
- **Fault tolerance**: This refers to the ability of a MANO to continue operating without interruption when it's components fail

s

2

Scalability Approaches

Some of the scaling approaches from various academic work are discussed in this chapter. In each section a brief introduction about each scalability approach and the relavance of a specific approach to our context of research (MANO scalability) has been discussed.

2.1 Service replication

A technique to clone services running on other nodes to stabilize the service load among different nodes without causing any damage to the ongoing operations. Services that are replicated secure additional resources provided by the new nodes for handling larger service load. In other words, service replication enhances service scalability and reduces the risk of QoS degradation by handling larger service loads. In a case study, [FB] Falatah and Omar performed an analysis by varying the system load.Firstly, a variable for service load is set. The service load is a number of service invocations within a unit time. For the case study, the unit time was set to be 500ms. That is, if ten invocations occur within 500ms, then the service load is 10. To show an effectiveness of service replication, they simulate the service replication scheme for seventeen different volumes of service load. On each service load, conventional service system is compared with service replication strategy in terms of average response time.

In terms of MANO, service replication is easier when the server is stateless, but the MANO will have a database, user session and various managed services. Simply replicating servers cannot be the solution, multiple MANO using a single database will lead to a bottleneck. Hence database clustering or database replication is also needed while maintaining uniformity accross the databases. Service replication increases availability and parallelism.

2.2 Service Migration

Service migration is a strategy of placing a service to a different node when a particular node cannot provide high QoS due to a hardware/software problem, or the physical distance between consumers and providers. After service migration, the migrated service performs the same role that it was supposed to performed on the unstable node and the unstable node is removed from the list of service nodes. The removal of this unstable node reduces overall QoS degradation [LK]. The above authors conducted a simulation where the service was migrated to a node that is located closer to consumer. There is also an assumption made that the response time is

directly proportional to the distance. Hence, a service is migrated to the fastest node in terms of response time.

In terms of MANO, they typically manage VIMs. There is a close association between MANOs and VIMs to get an NS instatiated. Mere migration of a MANO server closer to the user will not make the communication time between MANO and VIM faster. Therefore, in MANOs case, it seems like service replication is a better strategy.

2.3 Proactive scaling

Proactive scaling also known as scheduled scaling is mostly done in a cloud by scaling at predictable, fixed intervals or when big traffic stream is anticipated. A well-designed proactive scaling system enables providers to schedule capacity alterations based on a plan. With scheduled scaling, one can set when to increase the capacity or number of servers and when to decrease them. To implement proactive scaling, one should first understand expected traffic flow, which means the providers should have some kind of statistics which indicates the desired(usual) traffic, deviation (usually high) from expected traffic [FB][Ree].This type of scaling is suitable for servers that will have increased load during known days

In MANOs' terms, the MANO should be able to use these statistics and decide a scaling action. At the specified times, it should scale up with the values for normal, minimum or maximum traffic surges.

2.4 Reactive scaling

A reactive scaling strategy also known as auto-scaling adjusts its capacity by adding or removing, scaling up or down resources. This type of scheme could be useful when the scheduled scaling plan goes wrong in proactive scaling. Cloud providers as well as cloud agencies require periodic acquisition of performance data for maintaining QoS. In addition, reactive scaling enables a provider to react quickly to unexpected demand. The crudest form of reactive scaling is utilization-based i.e. when CPU, RAM or some other resource reaches a certain level of utilization, the provider adds more of that resource to the environment[FB][Ree]. This type of scaling is suitable for servers that will have increased load during few unknown days.

In MANOs' terms, the plan is to develop a software called as "scalability manager" that could be installed in every MANO which helps scaling more children MANOs. It is the responsibilty of the scalability manager to scale MANOs and redirect the requests to the child MANO. This is done when the NS instances that are assigned to a particular MANO reaches the threshold.

2.5 Predictive scaling

This type of scaling uses machine learning to predict the traffic stream of a server/application beforehand so that the capacity changes can be done accordingly, It collects data from all the VM instances and various other data points and uses well trained machine learning models to actually predict the flow of traffic. This model would make use of one day's data and then the data is re-evaluated every 24 hours. This type of scaling strategy will be useful where servers are affected with cyclic periodic loads.

In terms of MANO, this type of scaling uses data from MANO instances and predicts the load on the server with the help of a machine learning model. This strategy is yet to be explored more to incorporate this in MANO.

2.6 Service System Scaling - Dynamic node scaling

To scale a service system across different nodes where these nodes are distributed all over the Internet, management components which help in monitoring the status of all these nodes are needed. These components help consumers manage the nodes as consumers would be unaware of the physical location of a service node that they use. Therefore, [LK] propose two key components to manage service scalability such as Global Scalability Manager (GSM) and Regional Scalability Manager (RSM).

The key role of GSM is to manage service scalability. It balances service load in the system by obtaining the current status of nodes that are listed in the service system and designs a scalability assuring method. Depending on this design, it directs any functionality.

RSM component is installed on all the service nodes. It observes the status of its node and transfers it to GSM. It also executes the scalability assuring scheme based on the instructions from GSM. These two components assist the service system to add or remove new nodes at run-time called as dynamic node management.

Service Scalability Assuring Process

- In this process, firstly define metrics for scalability measure. This metric could also be used to decide raw data and to compute scalability.
- Certain techniques from Qos monitored services are used to collect the set of raw data items.
- Next step is to figure out scalability metrics. If the metrics indicate an acceptable scalability level then continue step 2 and 3 where as if the metrics indicate a need to repair the below averaged scalalability then execute the following steps.
- Next up is to develop a remedy plan for improving the below avaraged scalability considering the current states of monitored service. Scalability assuring strategies like service replication and/or service migration could be adopted depending on the complexity and nature of the suffered service.
- In this step, the selected scalability strategy is executed and this is quite often an automated process.
- The final step is to inspect the result of the remedy plan and understand from the entire procedure as to how to improve the scalability. If the outcome of the process is valuable, then both the consequence and the remedy plan are logged for future uses thus making it a smart scalability framework.

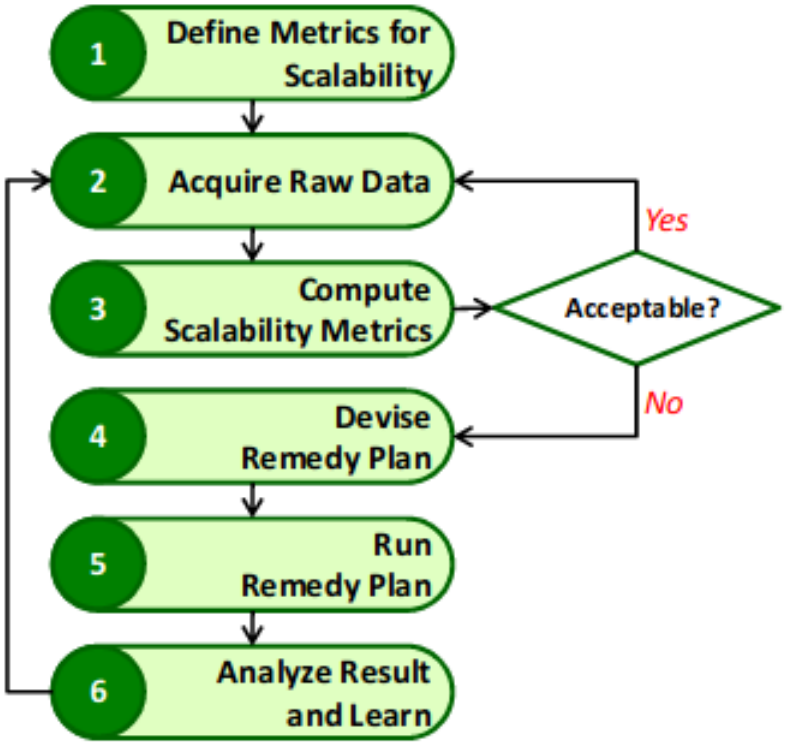

Figure 2.1: Service scalability Assuring Process from [LK]

2.7 Hierarchical Service Placement

In the centralized model, the service orchestrator has a detailed view of all Execution Zones(EZs).It may be impractical, for scalability reasons, for a globally centralised placement algorithm to maintain detailed knowledge of all users and all EZs and so here the author investigates a hierarchical solution where the overall orchestration domain is split into geographical sub-domains. In this model the high-level orchestrator has limited visibility of Execution Zone(EZ - Services will be deployed in datacenters/clouds called EZ) and user demands within a sub-domain - it sees only the aggregate of user demands and the aggregate of EZ capacities within a particular sub-domain. The high-level orchestrator places service instances at the coarse granularity of sub-domain only and subsequently each sub-domain orchestrator undertakes a further placement algorithm with the scope of that sub-domain only to determine in which specific EZs what quantity of service instances should be placed to supply the required number of session slots to meet the specific detailed demand pattern of user requests within that sub-domain.

There are many ways of sub-dividing an overall orchestration domain into sub-domains. One option is to map sub-domains onto the same geographical area covered by resolution domains: the entity responsible for resolving user requests to EZs with available session slots. Equating sub-domains for orchestration and service placement purposes with resolution domains is not essential as other coarser or finer grained sub-domains could be considered [MPGR].

3 Effects of scaling

3.1 Availability

Availability describes how often a service can be used over a defined period of time.

How to estimate the availability of a system

Most service outages are the result of misbehaving equipment. These outages can be prolonged by misdiagnosis of the problem and other mistakes in responding to the outage in question. Determining expected availability as stated in [Ree] involves two variables:

1. The likelihood that one will encounter a failure in the system during the measurement period.

2. How much downtime is expected in the event the system fails.The mathematical formulation of the availability of a component is:

$$a = (p - (c * d))/p \tag{3.1}$$

where a = expected availability
c = the % of likelihood that there is a server loss in a given period
d = expected downtime from the loss of the server
p = the measurement period

To achieve an accurate availability rating, one needs to rate all of the points of failure of the system and add them together. The availability of a system is the total time of a period minus the sum of all expected downtime during that period, all divided by the total time in the period:

$$a = (p - SUM(c1 * d1 : cn * dn))/p \tag{3.2}$$

3.2 Reliability

Reliability is [Ree] often related to availability, but it's a slightly different concept. Specifically,reliability refers to how well one can trust a system to protect data integrity and execute

its transactions. The instability associated with low availability often has the side effect of making people not trust that their last request actually executed, which can cause data corruption in relational database systems.

Much of the reliability of a system depends on how one writes the code that runs it. The cloud presents a few issues outside the scope of the application code that can impact a system's reliability. Within the cloud, the most significant of these issues is how persistent data is managed. Virtual instances tend to have lower availability than their physical counterparts, the chance for data corruption is higher in the cloud than it is in a traditional data center. In particular, any time one loses a server, the following factors become real concerns:

1. Loss of data stored on that instance which has not been backed up somewhere.

2. Block storage devices have a chance of becoming corrupted (just as they would in a traditional data center)

3.3 Heterogeneity

Heterogeneity refers to the state of being diverse. The scaling in a distributed system is also affected by the heterogeneity of systems involved. The administrative dimension of the scaling constitutes to the problem regarding heterogeneity focusing of both hardware and also software required, to deliver the services efficiently. One of the solutions to such a problem is coherence. In a coherence system, the different administrative systems have a common interface [oN94].

Administration in a MANO framework

The administrative domain in an NFV architectural framework is majorly divided into Infrastructure domain and Tenant domain. Infrastructure domains are defined based on the criteria like type of resource such as networking, compute and storage in traditional data-centre environments, by geographical locations or by organisation. The tenant domains are defined based on the criteria like by the type of network service, etc. In a framework, multiple infrastructure domains may co-exist, providing infrastructure to a single or multiple tenant domain. The VNFs and Network Services reside in the tenant domain which consumes resources from one or more infrastructure domains [PTM$^+$].

Multi-MANO Interworking

To achieve a better provisioning of network services, two or more Service Platforms (SPs) cooperate or one orchestrator leverages on the NFV interface or on the other orchestrator to instantiate functions, services. The infrastructure domain is segmented to accommodate the demands of separate organisation hence deploying a hierarchy of service platforms that need to collaborate in order to deploy NFV end-to-end services.The interaction between the two MANOs is achieved by mapping the services and infrastructure domains of the MANO.

In a hierarchical placement of the two MANO service platforms, it either supports complete outsourcing of a network service for deployment in a lower service platform or split the service deployment across two MANO SPs.Hence, the NFVO of the upper MANO constitutes a resource orchestrator (RO) along with network service orchestrator (NSO) to facilitate the services.

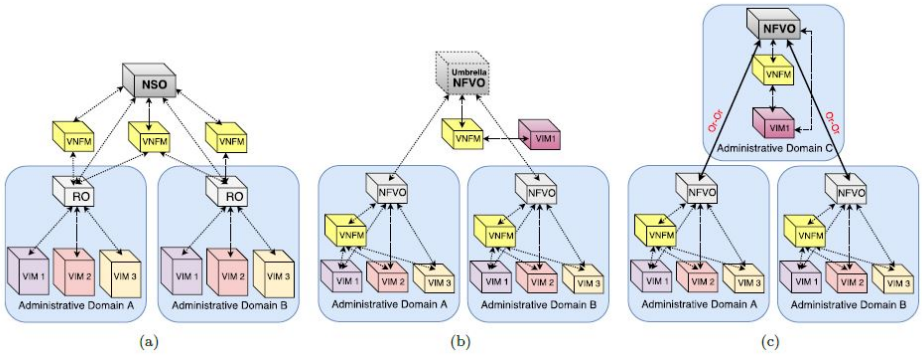

Figure 3.1: ETSI approaches for multiple administrative domains. Adapted from [dSPR+18]

According to [dSPR+18] , For a network service to support across multiple administrative domains, they require coordination and synchronisation between multiple involved infrastructure domains which are performed by one or more orchestrators. The ETSI approaches for multiple administrative domains are depicted in the figure below.

In the above figure 3.1, (a) refers to a approach in which the orchestrator is split into two components (NSO and RO), (b) refers to a approach with multiple orchestrators and a new reference point: Umbrella NFVO and (c) refers to a approach that introduces hierarchy and the new reference point Or-Or.

4

Scaling a Network Service

The Scaling of a network service plays a key role while handling the system load on a MANO or for a better performance. The network service contains a NSD that limits the instantiation levels of an NS instance, by defining them as the discrete set of levels addressing the number of instantiation levels required while scaling a network service [AHOLA+18].

According to [AHOLA+18],the deployment flavors of an NSD contains information about the instantiation levels permitted for an NS instance, with the help of information from VNFDs and VLDs. The VNF flavour of the VNFD specifies which part of VNFCs are to be deployed. The NS flavour selects the part of VNFs and VLs to be deployed as a part of NS. With this information it defines the instantiation levels.The below figure shows the scaling attributes of an NSD.

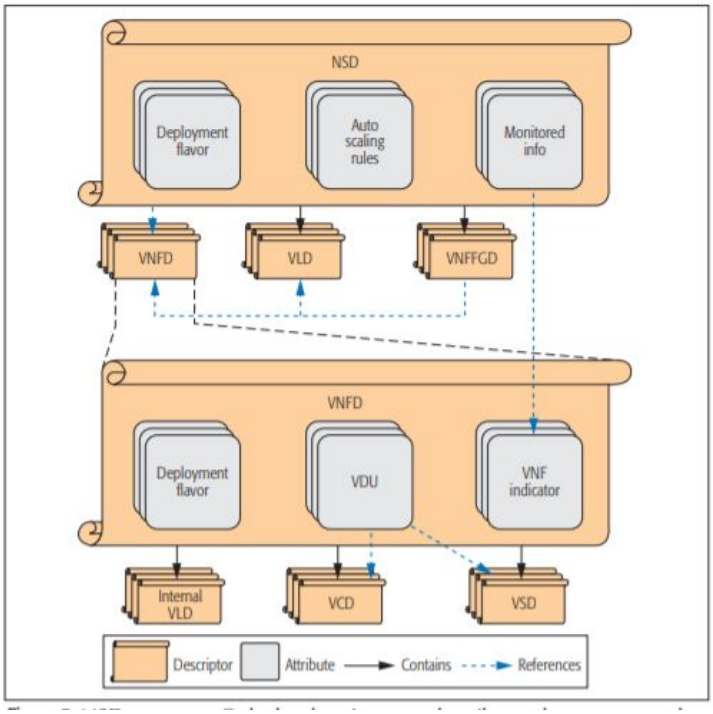

Figure 4.1: NSD structure. Adapted from [AHOLA+18]

5

Capacity of NFVOs and VNFMs

In distributed NFVI environment or when network services are spanned over large geographical areas, the number of VNF instances are likely to increase. Hence, there should be adequate number of VNFMs and NFVOs for manage the VNF instances. To determine the capacity of NFVO and VNFM, the Integer Linear Programming (ILP) formula is proposed [ALNGT17].

The below formula determines the optimal number of NFVOs and VNFMs required in a distributed system [ALNGT17].

> Consider the NFVI modeled as a graph $G = (P, E)$ where P is the set of NFVI-PoP nodes and E is the set of edges linking them, such that $E = \{(p, q) \mid p \in P, q \in P, p \neq q\}$. We use $\delta_{p,q}$ to represent the network delay of an edge $(p, q) \in E$. Let V represent the set of VNF instances in the system. The location of a VNF instance $v \in V$ is defined by $l_{v,p} \in \{0, 1\}$ such that $l_{v,p}$ equals to 1 only when v is placed at $p \in P$. We define M to represent the set of VNFMs that can be used to manage the VNF instances. We also use φ to denote the capacity of a VNFM. It represents the maximum number of VNF instances that can be managed by a VNFM.

Decision Variables:

$h_p \in \{0, 1\}$: (1) indicates that a NFVO is placed at $p \in P$, (0) otherwise.

$r_{q,p} \in \{0, 1\}$: (1) specifies that $q \in P$ is assigned to the NFVO which is placed at $p \in P$, (0) otherwise.

$x_{m,p} \in \{0, 1\}$: (1) designates that $m \in M$ is placed at $p \in P$, (0) otherwise.

$y_{v,m,p} \in \{0, 1\}$: (1) indicates that $v \in V$ is assigned to $m \in M$ which is placed at $p \in P$, (0) otherwise.

Mathematical Model:

$$\textit{Minimize} \quad \sum_{p \in P} h_p + \sum_{m \in M} \sum_{p \in P} x_{m,p} \tag{1}$$

Subject to:

$$\sum_{p \in P} r_{q,p} = 1, \quad \forall q \in P \tag{2}$$

$$r_{q,p} \leq h_p, \quad \forall q, p \in P \tag{3}$$

$$r_{p,p} = h_p, \quad \forall p \in P \tag{4}$$

$$\sum_{p \in P} x_{m,p} \leq 1, \quad \forall m \in M \tag{5}$$

$$\sum_{m \in M} \sum_{p \in P} y_{v,m,p} = 1, \quad \forall v \in V \tag{6}$$

$$y_{v,m,p} \leq x_{m,p}, \quad \forall v \in V, m \in M, p \in P \tag{7}$$

$$l_{v,q}\, y_{v,m,\acute{p}}\, r_{\acute{p},p} \leq r_{q,p}, \quad \forall v \in V, m \in M, q, \acute{p}, p \in P \tag{8}$$

$$\sum_{v \in V} \sum_{m \in M} \sum_{q \in P} y_{v,m,q}\, r_{q,p} \leq \Phi\, h_p, \quad \forall p \in P \tag{9}$$

$$\sum_{v \in V} y_{v,m,p} \leq \varphi\, x_{m,p}, \quad \forall m \in M, p \in P \tag{10}$$

6 Conclusion

The main goal of this report is to aggregate all the factors affecting the scalability of a MANO generally. Scalability is an important factor in cloud environments which accommodates the demanding needs and also not affecting the system's performance. The report further states the system load in terms of the load on the NFVO of a MANO. The scaling in a distributed system is affected by the availability of a service and its reliable quotient. The report further describes the effect of diverse administrative domains in a MANO framework and how it can be addressed by a coherence mean. Listed below are few of the scalability approaches which are aggregated from various research papers to tackle the challenges existing.

1. Service Replication
2. Service Migration
3. Proactive Scaling
4. Hierarchical Service Placement
5. Reactive Scaling, and
6. Service System Scaling

Further, by choosing one of the approaches and addressing the other factors, the research will continue.

Bibliography

[AHOLA+18] Oscar Adamuz-Hinojosa, Jose Ordonez-Lucena, Pablo Ameigeiras, Juan J Ramos-Munoz, Diego Lopez, and Jesus Folgueira. Automated network service scaling in nfv: Concepts, mechanisms and scaling workflow. *IEEE Communications Magazine*, 56(7):162–169, 2018.

[ALNGT17] Mohammad Abu-Lebdeh, Diala Naboulsi, Roch Glitho, and Constant Wette Tchouati. Nfv orchestrator placement for geo-distributed systems. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, pages 1–5. IEEE, 2017.

[AS08] Natee Artaiam and Twittie Senivongse. Enhancing service-side qos monitoring for web services. *2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pages 765–770, 2008.

[CMK] Trieu C. Chieu, Ajay Mohindra, and Alexei A. Karve. Scalability and performance of web applications in a compute cloud. In *2011 IEEE 8th International Conference on e-Business Engineering*, pages 317–323. IEEE.

[dSPR+18] Nathan F Saraiva de Sousa, Danny A Lachos Perez, Raphael V Rosa, Mateus AS Santos, and Christian Esteve Rothenberg. Network service orchestration: A survey. *arXiv preprint arXiv:1803.06596*, 2018.

[FB] Maram Mohammed Falatah and Omar Abdullah Batarfi. Cloud scalability considerations. 5(4):37–47.

[fur] Handbook of cloud computing. OCLC: ocn639164885.

[JW] P. Jogalekar and M. Woodside. Evaluating the scalability of distributed systems. 11(6):589–603.

[LK] J. Y. Lee and S. D. Kim. Software approaches to assuring high scalability in cloud computing. In *2010 IEEE 7th International Conference on E-Business Engineering*, pages 300–306.

[MPGR] E. Maini, T. K. Phan, D. Griffin, and M. Rio. Hierarchical service placement for demanding applications. In *2016 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6.

[noa] Scale in distributed systems(clifford) | scalability | server (computing).

[oN94] B Cli ord Neuman. Scale in distributed systems. *ISI/USC*, 1994.

[PTM+] Xiaoyan Pei, Deutsche Telekom, Klaus Martiny, NTT DOCOMO, Kazuaki Obana, António Gamelas, SK Telecom, and DK Lee. Network functions virtualisation (nfv).

[Ree] George Reese. Cloud application architectures. page 206.

[STCP17] Thomas Soenen, Wouter Tavernier, Didier Colle, and Mario Pickavet. Optimising microservice-based reliable nfv management & orchestration architectures. In *2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 1–7. IEEE, 2017.

[ZLC] Liangzhao Zeng, Hui Lei, and Henry Chang. Monitoring the QoS for web services. In Bernd J. Krämer, Kwei-Jay Lin, and Priya Narasimhan, editors, *Service-Oriented Computing – ICSOC 2007*, volume 4749, pages 132–144. Springer Berlin Heidelberg.