# Scalability Testing of MS Lync Services: Towards Optimal Provisioning of Virtualised Hardware

Knut Helge Rygg
IDI, NTNU
Trondheim, Norway
khsrygg@gmail.com

Gunnar Brataas
SINTEF ICT and IDI, NTNU
Trondheim, Norway
Gunnar.Brataas@sintef.no

Geir Millstein
Telenor GID
Fornebu, Oslo, Norway
Geir.Millstein@telenor.com

Terje Molle
Telenor GID
Fornebu, Oslo, Norway
Terje.Molle@telenor.com

## ABSTRACT

A method for scalability testing of the Microsoft Lync 2010 communication system is presented, exploring the relation between system size and system load. The method can be used for optimal provisioning, balancing user Quality of Experience (QoE) with equipment volume and energy consumption. Observing a standard edition of Lync, on a virtualised platform using the VMware hypervisor, the method indicated linear scalability. QoE was mainly limited by the Mean Opinion Score (MOS). This MOS limit corresponded to a Lync front end server utilisation of about 60%.

## Categories and Subject Descriptors

K.6.2 [**Computing Milieux**]: Performance and Usage Management; H.1.0 [**Information Systems**]: Models and Principles, General

## Keywords

Scalability, testing, virtualisation, provisioning, energy efficiency, MS Lync, workload, performance, data centres, telecom industry

## 1. INTRODUCTION

The objective of this paper is to describe a method for scalability testing of the Microsoft Lync 2010 communication system. In this context, scalability describes the relation between system load and required system resources. When we know the required system load, our method can find the required amount of system resources. Although the word "scalability" is often used, it still has no formal definition [8]. Therefore, methods for scalability testing have not received much attention.


*ICPE'13,* April 21–24, 2013, Prague, Czech Republic.


Scalability testing may be used as part of optimising provisioning processes. With optimal provisioning, we strike a balance between under- and over-provisioning of computer resources:

- With resource *under-provisioning*, Quality of Experience (QoE) will degrade, or we may even have outages.
- *Over-provisioning* gives low utilisation as well as high equipment, operational and energy costs.

In the ideal case, energy consumption should be proportional to the computer system utilisation [6], but until we reach this ideal, optimal provisioning is important. Even in a virtualised environment, it is beneficial to have provisioning guidance, which helps buying the right type and amount of hardware, by the right time.

This work was performed for Telenor Norway, the largest Norwegian telecom operator and part of the Telenor international operations. Microsoft Lync can be configured in various setups with different services and user scenarios. This paper focuses on *hosted Lync* as it is provided by Telenor. This means that Telenor hosts Lync servers on behalf of customers who do not have the resources to manage their own deployment. Telenor's requirements should also be representative for industry requirements in general.

We have studied the operation of Lync in a virtualised environment, using the VMware hypervisor. As a workload generator, we used the Lync Server 2010 Stress and Performance Tool [3].

This work was performed as a master's thesis [16], which imposed limitations in terms of time and available equipment. These limitations are also typical in industry.

This paper describes related work in Section 2. The scalability baseline is outlined in Section 3. Our method is described in Section 4 and applied to Lync in Section 5. In Section 6, we offer conclusions and outline further work.

## 2. RELATED WORK

In this description of related work we focus on the issue of Lync provisioning. The Lync Planning Guide by Microsoft [13] is based on physical servers and is crude for virtualisation. The scenario-based capacity planning spreadsheet [15] (also from Microsoft) is a more fine-grained analysis of Lync capacity requirements. The system designer can

specify both the number of users and a custom user model by varying a number of different parameters. The output is a recommendation on the number of servers to use and their respective CPU utilisation, memory usage, etc. The calculations are still based on the same standard configuration hardware described in the capacity planning guide and the percentage of external users is hard coded to 30%. This does not fit with Telenor's hosted scenario where all users are external. The resource demand on the edge server will be higher than the calculator falsely indicates. Even though the capacity calculator allows more flexibility, it is not sufficient to provide good estimates for Telenor's deployments.

Although provisioning of Microsoft Lync is described in books explaining Microsoft Lync deployment [20, 11], they only reference Microsoft's own capacity planning guidelines. Therefore, there is little help for those planning to use custom hardware or specialised scenarios.

Bandwidth usage in Microsoft Office Communication Server (OCS) is described in [17]. OCS is a predecessor of Microsoft Lync with much of the same functionality, except for telephone integration. The overall server load in terms of network traffic can be estimated from the connected user endpoints and their respective traffic patterns. Peak network utilisation can be quite different from the average, so using mean values may be inaccurate for bandwidth provisioning. Network usage is proportional to the number of users. However, the report gives little information on provisioning other hardware resources like CPU, memory, or storage.

A sizing study of Microsoft Lync [18] describes a test environment built by Dell PowerEdge R720 servers using LyncPerfTool. The report finds that the number of users scales linearly to the system size in the range of 3,000 to 12,000 users. Compared with the Telenor deployment studied in this report, the differences are:

1. Dell's sizing study uses the enterprise edition of Lync, which allows several servers to be assigned to the same pool. The number of front end server virtual machines are 1, 2, and 4. For each configuration, the maximum number of users are found. Our paper uses the standard edition of Lync which does not support pools.

2. Dell's sizing study uses virtual machines for the front end servers, while the remaining servers are installed on physical machines. In this paper, all Lync servers and databases are virtualised.

3. Dell's sizing study has only internal users. Internal users do not require edge or reverse proxy servers. However, this paper is based on a hosted Lync scenario which has only external users. The edge server and reverse proxy server roles are therefore included in this paper.

# 3. SCALABILITY BASELINE

This section describes basic scalability concepts which are prerequisites for the method described in Section 4. We build on the scalability framework outlined by Brataas & Hughes et al. [7]. We first describe *workload*. Then we describe *performance* and finally *scalability*.

## 3.1 Workload = Work + Load

In our framework, we separate workload into work and load. *Work* is about what is done and is determined by the type and nature of the invoked services. A *system* consists of connected components. Each *component* in the system has one or more *services*, which are different types of work it offers to other components. *Load* describes how often the services are performed, and is measured in terms of throughput, i.e., the number of finished tasks/jobs during a given time interval, but can also be the number of users in a closed system. Work and load together constitute workload.

## 3.2 Performance

The performance of a system may describe the system quality as a function of system load as depicted in Figure 1. With higher load the quality degrade. Implicitly in this figure, work is kept constant. A common measure of system quality is response time. With increasing load, response times typically increase too. The capacity of the system is the load when a defined operating point is reached. This operating point may also be termed service level agreement (SLA) or quality limit.

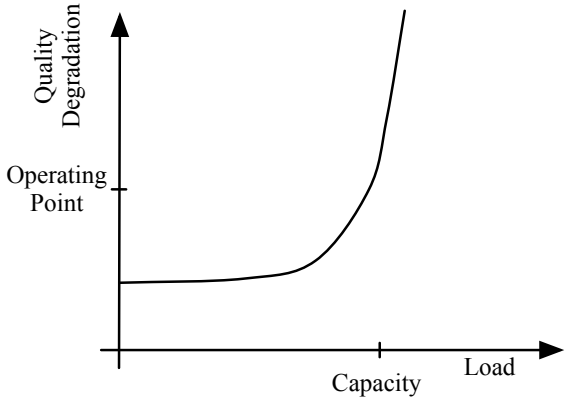


**Figure 1: Performance of a system.**

## 3.3 Scalability

System size is normally constant when measuring the performance of a system, but not when we describe the scalability of a system. Scalability is a relation between system size and system capacity, given fixed system quality and work, as described in Figure 2. If system capacity is proportional to system size and the quality of the system is constant, then we have linear scaling. This is not the case in Figure 2, where we clearly see that system capacity reach an upper limit. Further increase in system size actually leads to decreasing capacity.

The concept of range is also important. A system may be linearly scalable within a certain system size range. For example, there may be approaches with large scalability problems which do not matter if these problems only occur in ranges not used.

In our framework, system size has three dimensions, processing speed, storage amount, and connectivity, but in this work, we focus only on processing speed. System size is also hierarchic, and we can typically define the system level, the

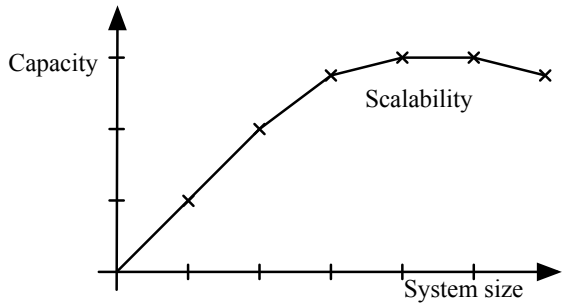


**Figure 2: Scalability of a system.**

subsystem level, and the device level. On all of these levels, size can be increased by replication. Replication at the subsystem level (adding more servers for each server role like the front end server) is typically called *scale out*, while *scale up* refers to replication at the device level (adding more CPUs, disks, etc.)

# 4. METHOD

The empirical scalability testing method for finding optimal provisioning builds on the scalability concepts in Section 3. The objective of the empirical tests is to determine the maximum number of users for each system size and user workload scenario. A number of quality metrics are used to assure that the quality degradation of the Lync services are within acceptable limits. The maximum number of users can be found by methods of systematic trial and error and adjusting the number of users for each run. The number of users is increased if quality metrics are within limits, and decreased if the limits are exceeded.

One way of reducing the search space is to use binary search. For each iteration of the binary search, the search space is halved. Interpolation on quality metrics could also be considered, but since there are no guarantees on linearity in the metrics, a binary search is a more robust alternative. The binary search can theoretically continue until the search space has only one remaining user. However, searching with such fine granularity would probably not produce stable results because of the random errors in all measures. Such errors are more dominant in fine granularity tests. Running many iterations also requires more time for testing. In these tests 4 - 6 iterations were used for all system sizes until an acceptable value was found.

*The following numbering refers to Section 5, where the method is applied to the Lync test case.*

**5.1: Understand environment** The method relies on a test bed, enabling the use and validation of the method, as well as introducing some limitations and constraints. The test bed has three parts: We start on the bottom with hardware and continue with software and load generation:

- **5.1.1: (Test bed) hardware** where we also outline the virtualisation platform.
- **5.1.2: (Test bed) software** which in our case describes the Lync server roles used.
- **5.1.3: (Test bed) load generation** used to generate test bed workload.

**5.2: Identify requirements** for scalability testing. There are several types of requirements concerning system size scaling path, work, and quality metrics:

- **5.2.1: System size scaling path** Explore the system size scaling options available and select an enumerated set of these options. Questions to be answered include: Where can we add resources in our configuration so that we can increase the size of the system? What are the practical resource ranges/limits? Which discrete set of resource quantities should be selected for further investigations?
- **5.2.2: Determine work** Which types of services are relevant? Work is further described in Section 3.1.
- **5.2.3: Explore the quality metrics** most applicable for our purposes. In our case, there are many, such as MOS (Mean Opinion Score), CPU utilisation, RTT (Round Trip Time), jitter, and packet loss. The task here is simply to get an overview of all of them.

**5.3: Run tests** to find the capacity for each system size as outlined in Section 3.2 and Section 3.3. In our case this step involves:

- **5.3.1: Tune configuration** e.g., allocate appropriate number of vCPUs to each of the Lync server roles.
- **5.3.2: Determine run length** For how long do we need to run each experiment?
- **5.3.3: Run experiments** using binary search for each system size.
- **5.3.4: Decide on quality metric** determining if a given load (capacity) is within our quality limits.
- **5.3.5: Find capacity for each system size**

**5.4: Analyse results** consisting of the following steps:

- **5.4.1: Find bottleneck resource** This gives insight into the system by finding the bottleneck resource, which is useful when we are upgrading the hardware. Monitoring the bottleneck resource is a good way to keep an eye on QoE.
- **5.4.2: Analyse the scalability** i.e., the relation between capacity and system size. Is it linear? In detail, this step involves finding the capacity for each system size using the same operating point. We use binary search to find the capacity for each system size.
- **5.4.3: Extrapolation** Is it possible to extrapolate beyond measured system size range?

Although this method appears to be linear (where we follow the steps in natural order), some degree of iteration (backtracking) will be required. For example, to determine the run length in step 5.3.2, we needed to know the number of IM (Instant Messaging) conference users, which was determined by the next step 5.3.3, running the experiments. In step 5.3.1, we first tried using the default value of 40 GB disk for the SQL server, but then the disk ran full in step 5.3.3, and we had to adjust the disk size to 100 GB.

# 5. APPLYING THE METHOD

In this section, we apply the method from Section 4 and describe the challenges and results for each step. In Section 5.5, we comment on the amount of human time required to perform these steps.

## 5.1 Understand Environment

This step focuses on understanding the test bed, which consists of virtualised hardware, Lync software, and a load generator.

### 5.1.1 Hardware & Virtualisation

In this section, the platform is described on a detailed level. Many of the details will not be essential for the upper layers of the test bed and the further steps of the method.

The hosted Lync deployment is built using components from Cisco Unified Computing System [5] (UCS) architecture. The architecture consists of blade servers that are connected to a Cisco UCS 5108 blade server chassis. A blade server has 96 GB RAM, two Intel Xeon E5650 CPUs, each with 6 cores and hyper-threading, and two 15k rpm SAS disk drives. Each blade server chassis is connected to a 10 Gbps fabric interconnect using a Cisco UCS 2104xp fabric extender. A Cisco UCS 6140 fabric interconnect can be connected to a SAN to provide increased storage for the blade servers. The architecture supports up to 40 blade server chassis on the same fabric interconnect. Each blade server chassis can take up to 8 blade servers. This allows for 320 blade servers in the same infrastructure. A central RAID60 SAN is connected to the fabric interconnect. The SAN is divided into disk groups consisting of 20 disks.

Intel Xeon has 9 discrete power states that can be selected in order to lower processor speed and power usage. The VMware ESXi hypervisor 5.0 has four different host power management policies to choose from. We used a high performance policy where the processor always runs in the highest processor state.

The VMware hypervisor is used as a virtualisation platform. Each blade server (also called host) is running separate instances of ESXi 5.0. VMware vCenter is used to manage all ESXi hosts. Several hosts can be organised into resource pools containing one or more virtual machines for each host. VMs can move from host to host within the resource pool and automatic monitoring can be enabled to maintain resource balancing among hosts. VMs are then automatically migrated from a saturated host to a new host with enough available resources. Virtualisation can therefore provide flexibility along with better hardware utilisation.

Virtualisation has some advantages compared to bare machine performance. VMware ESXi provides a proprietary transparent page-sharing technique that detects identical memory pages used by several VMs. This proves useful when several VMs use the same OS, shared libraries, etc. Pages containing shared libraries and static OS components are normally locked to read-only mode and several VMs can therefore safely access them and avoid keeping their local copy. This allows for memory overcommitment where memory entitlement is larger than available memory.

However, virtualisation can also lead to performance degradation:

1. Even if hardware-assisted CPU virtualisation offers a guest mode, the processor must still exit guest mode and enter root mode, for example when updating page tables.
2. The VMkernel as well as the VMconsole requires CPU processing time.
3. When VMs use physical memory, a double lookup is needed. The OS has its own virtual memory which is mapped to (what the VM thinks is) physical memory. This must again be mapped to the actual machine memory. A shadow lookup table can be used to decrease lookup wait time, but at the cost of maintaining more lookup tables, which consumes RAM. Our Intel Xeon E5650 supports hardware-assisted memory virtualisation where two layers of page tables are supported. Still, the TLB miss latency is higher.
4. Both the VMkernel and the VMconsole, as well as each VM, need to allocate and use memory.
5. Most hypervisors allow virtual network components (hubs, switches, routers, etc.). These are emulated in software and require resources to run. The Cisco UCS [5] technology provides some virtual networking functionality that reduces the need for software emulation.

For more information on the performance impact of virtualisation, consult [9].

Even though this paper focuses on scalability in Microsoft Lync, studying the scalability of virtualised servers can prove useful in a broader perspective because the interest and usage of virtualisation is increasing.

### 5.1.2 Lync Server Roles

This section describes which server roles we focused on and why. In our experiments, a total of seven Lync servers were used, and together constitute what is called the central site.

- The *reverse proxy* allows external users to download content.
- The *edge server* allows external users to connect to internal servers.
- The (standard edition) *front end server* contains the back end server and the A/V (audio / video) server role.
- The *domain controller* is an active directory (AD) domain controller and Kerberos key distribution centre and authentication server (KDC/AS).
- The *monitoring server* collects data to monitor QoE.
- The *SQL server* is a database for the monitoring server.
- The *mediation server* enables telephone integration via a PSTN gateway.

In a *hosted Lync* deployment a service provider (like Telenor in this case) provides Lync functionality to customers who connect remotely. All servers are administered at a central site and the customers only need to install client software. This is a huge benefit for the customer and many small and mid-sized companies choose this solution when they start using Lync. Lync can be configured in a variety of ways, including other server roles and both internal and external

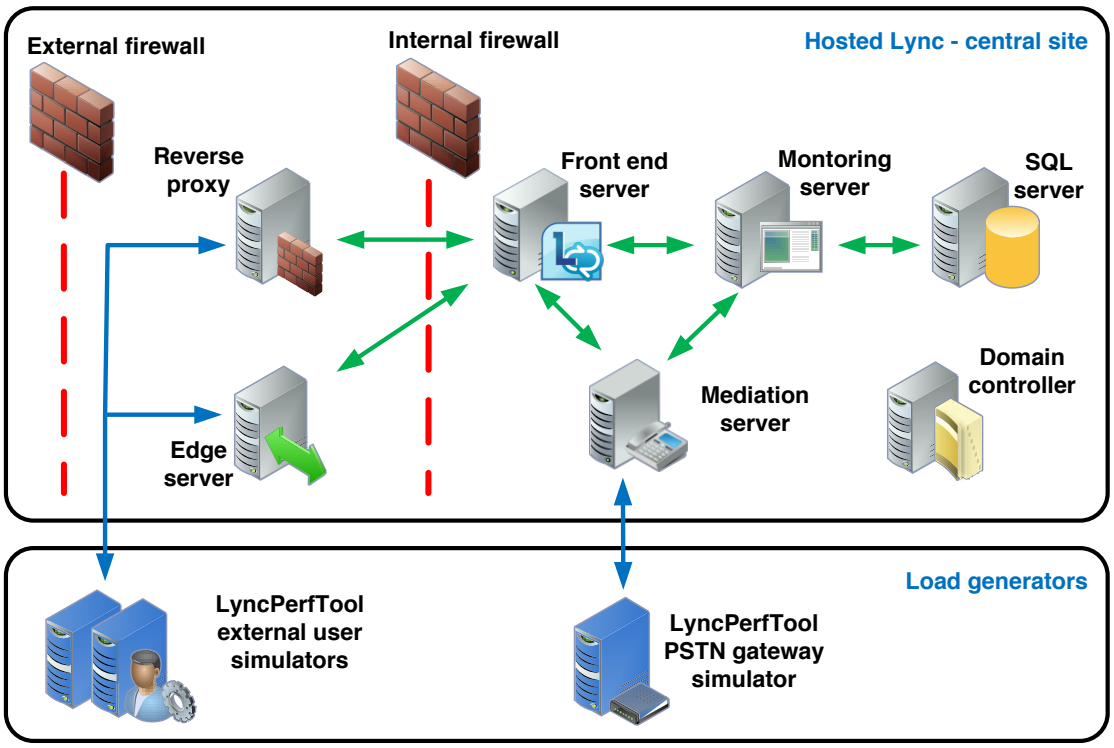


**Figure 3: Reference architecture for hosted Lync deployment.**

users, but the setup and workload in this test environment is typical for a hosted Lync deployment.

All of these seven servers are shown in Figure 3. The green arrows between the servers indicate which servers communicate with each other in order to process the workload imposed by users. The domain controller is not directly related to Lync, but more to Microsoft active directory. It communicates with all servers except the reverse proxy. The Kerberos part of the domain controller is essential for encrypted communication between servers, but the utilisation of the domain controller is hardly affected by the Lync workload (the total utilisation never exceeds 2%). Details on server-to-server communication can be found in [16].

All Lync servers are setup as VMs running on a blade server (Cisco UCS M200). The user simulators and the PSTN gateway simulator are running on two identical servers, but separate from the Lync server roles to avoid interfering with performance measures.

All VMs are provisioned with (more than) enough memory so it should be possible to keep a complete working copy of the VM image in-memory and occasionally write the changes back to the SAN. The local hard drives on each ESXi host contain swap memory which is used whenever the ESXi host runs out of memory. Esxtop [1] is a performance measurement tool that runs on each ESXi host. Results from this tool showed that swapping never occurred. Communication between the Lync servers is handled using a dedicated VLAN on the link to the fabric interconnect. The fabric interconnect uses 10.0 Gbit/s links and the latency is negligible. All Lync servers shared the same Microsoft Windows Server 2008 R2 operating system.

Other Lync roles like director and archiving were not used in our configuration. The director is an optional server role created to offload the internal servers by authenticating external user requests before sending them to other servers. This would give better performance if the director had dedicated hardware, but as long as the director and front end server share the same resources, there is no performance gain. The archiving server archives IM (Instant Messaging) content, but should only be used by companies that have legal compliance concerns.

In this test setup, user simulators, the PSTN gateway simulator, and the Lync server roles are all connected to the same LAN. In a real configuration, external users would be connected to the edge server and reverse proxy by an IPT connection (WAN). The telephone users would be connected by another IPT connection and a PSTN gateway. The PSTN gateway simulator used fewer resources than both of the two LyncPerfTool Clients, and was therefore put on the same physical server as one of them. As a result, we used three physical servers: one for all of the seven Lync servers, and the remaining two for the LyncPerfTool Client (with the PSTN gateway on one of the latter two servers). Firewalls are used to separate the internal Lync servers from the remaining network. External users can only access the edge server or reverse proxy server.

Our Cisco UCS platform used during measurement is similar to a production Cisco UCS platform with one small exception: The blade servers have a single network link to one fabric interconnect. In production, each blade server would have two links, each going to a different fabric interconnect for better redundancy. A single link in the test setup allowed a single point of monitoring, which was used for controlling the traffic among servers.

### 5.1.3 Load Generation

This section describes the load generation. Ideally, we would like to test each Lync service on its own and we would also like to characterise each Lync component individually to have a clear view of scalability. However, this would require developing a new load generator for Lync. It quickly became clear that this was not possible within the limited project resources. We therefore had to rely on the Lync Server 2010 Stress and Performance Tools [3] (LyncPerfTool).

The tests described in this paper used two Cisco UCS M200 blade servers for running the LyncPerfTool clients. The configuration is shown in Figure 3. There are three drawbacks with the test setup that may create bias in the performance measurements:

1. In a real deployment both clients and the PSTN gateway would be connected by an IPT connection, not by a LAN.
2. In a real deployment, users at a customer site may use wireless networks. Both IPT and a wireless network will introduce significantly more network delay and packet losses compared to a LAN.
3. In a real deployment, all of the network traffic can be (and normally is) encrypted. LyncPerfTool does not support encrypted protocols so in the test environment the traffic to and from the PSTN gateway simulator is unencrypted.

LyncPerfTool can simulate all of the main user workload types, except for these three: (1) group chat (2) video workload for peer-to-peer calls or conferencing, and (3) web conferencing. The group chat functionality enables multiple users to participate in conversations in which they post and access content about specific topics. Each session can be persistent. Enabling group chat requires installation of special group chat communications software and a back end database on separate servers. The group chat functionality is not a part of Telenor's standard hosted configuration. Therefore, the lack of support for this workload type in LyncPerfTool will not affect the measurements.

For handling video conferencing, there are three alternatives:

1. Direct address between the two Lync clients, which is the shortest and fastest option, if allowed by firewalls.
2. Connecting using NAT routers.
3. Relaying the video session through the edge server external network interface.

With only external users, the lack of video simulation in LyncPerfTool will only create a performance bias as long as we either use alternative 1 or 2, and not involve the edge server in alternative 3. In a hosted Lync environment, alternative 1 or 2 will be used. LyncPerfTool only supports web conferencing using the web access client, which does not allow voice or video in the conferences. However, both audio and video is simulated in dedicated audio/video conferences.

## 5.2 Identify Requirements

This section describes the requirements for the scalability testing. The requirements involve the system size scaling path, the work, and also the quality metrics.

**Table 1: Scaling hardware resources**

| Size | CPU | Network | Memory | Storage |
|---|---|---|---|---|
| 4/4 | 4/4 | 4/4 | 4/4 | 4/4 |
| 2/4 | 2/4 | 4/4 | 4/4 | 4/4 |
| 1/4 | 1/4* | 4/4 | 4/4 | 4/4 |

* The level 3 shared cache is scaled 2/4

### 5.2.1 System Size Scaling Path

We now have to consider where we can increase the system size. Since we studied a Lync deployment installed with a standard edition license, it is not meaningful to add more than one physical server for the edge server role (normally termed *scale out*). However, it is possible to add more resources to the physical server (normally termed *scale up*). The maximum system size was limited by the resources available. In this test setup, only three Cisco UCS 200 M2 blade servers were available for running the virtual Lync servers and load generators.

The load generators required twice the hardware resources of the Lync servers, so only one blade server was used for virtual Lync servers. The system size was therefore scaled by reducing the resources available on the blade server. The CPU resources can be scaled by disabling processors or processor cores. As mentioned in Section 5.1.1 the blade servers use two Intel Xeon E5650 with 6 cores and hyper-threading support. This gives a total of 24 virtual cores that are all enumerated by the hypervisor.

By using CPU affinity setting in VMware vSphere client, it is possible to decide which virtual cores can be used to schedule each vCPU of a VM. For the 2/4 system size, the affinity setting for all VMs was set to 0-11. This effectively disabled one of the two processors. For the 1/4 system size, the affinity was set to 0-5, so that only half of the cores of one processor could be used.

However, while the number of cores is constrained, the processor caches are still widely available. The level 1 and 2 caches are separate for each core, so these scale according to the number of cores, but the level 3 cache is shared among all 6 cores of one processor. This means that for the 1/4 system size, the number of cores is correctly scaled, but the level 3 cache is scaled 2/4, so it is twice as big as it should be. This gives a bias towards better performance estimates because more instructions and data can be stored in the level 3 cache instead of being fetched from main memory, which has a larger access time.

Network resources were not scaled in this test setup. Each virtual machine has access to the 10.0 Gbit/s network interface of the blade server. Still, most of the communication between VMs is internal to the server. Only the edge server, reverse proxy, and mediation server need to use the network interface.

Memory resources or storage resources were not scaled. Table 1 summarises the hardware scaling.

### 5.2.2 Determine Work

As described in Section 5.1.3, Lync has several types of services or work, but as also described in this section, we had to use the LyncPerfTool load generator. However, LyncPerfTool allows both light and heavy work, reflecting different user types. We therefore used both types of work to make our results more robust.

### 5.2.3 Explore Quality Metrics

Quality metrics were collected from two distinct sources, the monitoring server and the LyncPerfTool clients. The limits in Table 2 are found in Microsoft's recommendations [4] and the Lync sizing study [18]. The SIP limits are LyncPerfTool counters and the rest is monitoring server statistics.

**Table 2: Quality metric limits [4, 18]**

| Metric | Limit |
|---|---|
| Round trip time (RTT) | 100 ms |
| Jitter | 20 ms |
| Packet loss | 0.1 per sec. |
| MOS degradation | 0.5 |
| SIP 503 messages/sec | $\approx 0$ |
| SIP 504 messages/sec | $\approx 0$ |

The front end server and mediation server automatically sample all ongoing sessions and send CDRs to the monitoring server. The monitoring server produces statistics on audio quality using several metrics. Round trip time describes the time it takes from when a request is sent to when the answer is received. A high round trip time can be noticed by the user as a speech delay. Buffering enables acceptable speech quality even with a high round trip time. Jitter measures the variance in round trip time. A high jitter value makes buffering and processing hard because the packets arrive at highly variable times. Packet loss denotes the number of packets that are lost in transmission per second. The real time network protocols used in Lync have several built-in mechanisms for reconstructing the speech with acceptable quality despite some lost packets. However, when the packet loss exceeds 0.1, the packet loss affects call quality.

Mean opinion score (MOS) is a compound metric used to measure end user service quality of experience. MOS is a technically synthesised counterpart to the mean subjective rating of call quality on a scale from one (bad) to five (excellent) [10]. MOS is calculated using both transport layer parameters such as packet loss, jitter, and delay, as well as payload parameters such as audio codec, noise-level, echo, gain, and talk-over effects [12]. Each codec that is used in Lync has a maximum MOS value [2]. The monitoring server records the degradation of MOS value, i.e., the difference between the maximum MOS and measured MOS. The MOS degradation should not exceed the 0.5 limit. There are actually three MOS values, one for conference calls, another for peer-to-peer calls, and then a final one for PSTN calls. We are interested in the maximum degradation, and will therefore use the highest MOS degradation value.

SIP 503 is a reply message indicating that the Lync server is too busy to handle the request. A SIP 504 message is a server timeout message. LyncPerfTool counts the average number of such messages per second. Having some messages is acceptable, but over a longer period of time, the number should be close to zero.

A number of quality metrics could also have been collected from the front end -, edge - or mediation servers. This instrumentation would, however, affect the performance measures and are not included in the tests. The monitoring server reports were collected after each test run and will not affect performance. The metrics collected on the LyncPerfTool clients also did not create any measurement bias.

**Table 3: VM resource allocation**

| | | | vCPU/syst. size | | |
|---|---|---|---|---|---|
| **VM** | **Disk** | **RAM** | **1/4** | **2/4** | **4/4** |
| Front end | 40 GB | 32 GB | 4 | 7 | 14 |
| Edge | 40 GB | 16 GB | 2 | 4 | 7 |
| SQL | 100 GB | 32 GB | 1 | 2 | 3 |
| Mediation | 40 GB | 16 GB | 1 | 1 | 2 |
| Monitoring | 40 GB | 16 GB | 1 | 1 | 1 |
| Reverse proxy | 40 GB | 16 GB | 1 | 1 | 1 |
| Domain contr. | 40 GB | 16 GB | 1 | 1 | 1 |
| SUM | 340 GB | 144 GB | 11 | 17 | 29 |

## 5.3 Run Tests

The overall aim of this step was to find the capacity for each system size in the scaling path. This step consists of several sub-steps described in detail below.

### 5.3.1 Tuning Virtual Resources

When deploying virtual machines (VMs), it is important to make sure each virtual machine has the resources it needs to operate well. An overview of resource allocations for each virtual machine (VM) is shown in Table 3. The disk drive size of most virtual machines is 40 GB, which is the default allocation size for Microsoft Server 2008 R2 installations. For the SQL server, 100 GB was required to handle all of the call detail records generated for each ongoing audio session. Memory is provisioned to each virtual machine based on the hardware requirements for similar physical machine configuration recommendations from Microsoft [14].

There are several allocation options for provisioning CPU resources. The VMs in the test setup were provisioned using *allocation shares*, in contrast to *allocation reservation* or *allocation limit*, because shares are more flexible, allowing the VMs to share unused resources. Moreover, using allocation shares was also the preferred allocation option in the production environment in Telenor. Using allocation shares, each VM had a normal allocation setting with 1000 shares per vCPU (in contrast to the high value of 2000 shares or the low value of 500 shares), so the actual CPU allocation was defined by the number of vCPUs allocated to each VM. It is worth noting that all of the VMs had the same allocation priority. Therefore, the priority of each VM was determined by the number of vCPUs only. The number of vCPUs that were allocated for each VM was calculated based on the utilisation of each VM, $U_{\text{VM}}$, and the number of available virtual processors $U_{\text{total}}$:

$$\text{vCPU}_{\text{VM}} = \lceil \frac{U_{\text{VM}}}{U_{\text{total}}} \times N_{\text{logproc}} \rceil$$

As shown in the previous section, the number of available logical processors, $N_{\text{logproc}}$, is 6 for 1/4 system size, 12 for 2/4 and 24 for 4/4. The $\lceil$ and $\rceil$ delimiters in the formula simply mean rounding up to the next whole integer. Each server is provisioned with its relative share of available vCPUs. The rounding up makes sure that no server gets less than its relative share, but this means that the total number of allocated vCPUs will be a bit larger than the available logical processors.

Theoretically, each VM could have allocated the maximum number of vCPUs. However, over-provisioning CPU resources may degrade performance. The VMware CPU

scheduler uses an algorithm called relaxed co-scheduling [19]. Co-scheduling means that all vCPUs belonging to the same VM should be treated as a group. Therefore, the hypervisor tries to schedule all vCPUs in a group at the same time. Non-continuous scheduling of vCPUs of the VMs would create a different processor experience than on a physical processor. SMP programs that use several threads to complete a compute-intensive task may rely on synchronisation among threads. If some threads are halted because a vCPU is not scheduled, all other threads (and their respective vCPUs) will suffer a delay. The VMware hypervisor therefore implements a co-scheduling algorithm that tries to assign equal time intervals to each vCPU. If a certain vCPU has received more processing time than the other vCPUs in the same group, it is put into a waiting state called CO-WAIT. *Relaxed* co-scheduling means that a vCPU must succeed before a certain time limit in order to be put in the CO-WAIT state. More vCPUs per group increases the chances of one or more vCPUs being put into the CO-WAIT state. Spending time in the CO-WAIT state decreases performance. The vCPU calculation method described above will reduce the CO-WAIT overhead. This method is not limited to this deployment, but can prove useful in other multi-VM environments.

### 5.3.2 Determine Run Length

LyncPerfTool first signs in all users at a rate of 1 user per second. The different workload types are run as separate processes, so several users are signed in in parallel. The IM (Instant Messaging) conferencing workload process had the most users, with 1,587 IM users with light work out of a total of 4,800 users. All IM users are signed in after 26 min and 27 seconds. A transient interval of 30 min was therefore selected. Some of the tests could have used a shorter transient time, but standardised transient times made log parsing easier. We used a steady-state measurement period of 60 min, so that each test run took 90 minutes. The esxtool tool measured the performance of each VM in isolation from the hypervisor overhead.

### 5.3.3 Run Experiments

Experiments were run using the binary search protocol outlined in the Section 4. As described in Section 5.2.1, we decided to explore the system sizes 1/4, 2/4 and 4/4. Furthermore, we decided to explore both heavy and light work as described in Section 5.2.2. The complete results, which cover several pages, are presented in [16]. An overview of all the test runs is shown in Table 4 and in Table 5, for heavy and light work, respectively. In these tables, each test run has one row. The first column is the system size and the second column is the number of users. The third column is the % utilisation of the front end server divided by the allocated number of vCPUs, and the fourth column is the same for the edge server. We focus on the front end and edge server since they had the highest utilisation. The fifth column is the maximum MOS degradation.

The monitoring server measures MOS for (1) conferences, (2) peer-to-peer calls, and (3) PSTN calls. They all have the same limit of 0.5 MOS degradation. Therefore, we are only interested in the maximum of the three. From Table 4 and Table 5, we also see how MOS behaves with both lower and higher load than what gives the optimal MOS degradation.

We have only run four test runs for system size 1/4 with heavy work. Considering the MOS for 350 users we see that this value is exactly the limit of 0.50, and more runs are therefore not necessary.

**Table 4: Test runs for heavy work.**

| *Size* | *# Users* | $U_{front}$ | $U_{edge}$ | *MOS* |
|---|---|---|---|---|
| 1/4 | 200 | 34 | 28 | 0.10 |
| | 300 | 48 | 43 | 0.28 |
| | 350 | 58 | 52 | 0.50 |
| | 400 | 74 | 62 | 0.63 |
| 2/4 | 400 | 37 | 32 | 0.06 |
| | 600 | 53 | 44 | 0.13 |
| | 650 | 59 | 50 | 0.45 |
| | 700 | 65 | 56 | 0.64 |
| | 800 | 84 | 69 | 0.78 |
| 4/4 | 800 | 42 | 35 | 0.08 |
| | 1200 | 56 | 49 | 0.30 |
| | 1250 | 59 | 52 | 0.44 |
| | 1300 | 64 | 54 | 0.55 |
| | 1400 | 75 | 62 | 0.78 |
| | 1600 | 97 | 85 | 0.82 |

**Table 5: Test runs for light work.**

| *Size* | *# Users* | $U_{front}$ | $U_{edge}$ | *MOS* |
|---|---|---|---|---|
| 1/4 | 800 | 48 | 42 | 0.21 |
| | 1000 | 59 | 52 | 0.43 |
| | 1100 | 67 | 57 | 0.52 |
| | 1200 | 79 | 65 | 0.59 |
| | 1600 | 99 | 90 | 0.69 |
| 2/4 | 1200 | 39 | 32 | 0.07 |
| | 1800 | 53 | 46 | 0.24 |
| | 2000 | 59 | 51 | 0.47 |
| | 2200 | 68 | 56 | 0.59 |
| | 2400 | 80 | 63 | 0.78 |
| 4/4 | 2400 | 37 | 33 | 0.09 |
| | 3600 | 55 | 49 | 0.26 |
| | 3900 | 60 | 53 | 0.47 |
| | 4050 | 63 | 55 | 0.52 |
| | 4200 | 69 | 60 | 0.69 |
| | 4800 | 85 | 74 | 0.76 |

### 5.3.4 Decide on Quality Metric

All quality metrics have different limits and units. In Figure 4, all parameters from the monitoring server are normalised by dividing the measured value by the limit. This means that when the curve crosses the horizontal bold line, the limit is reached. The results come from the 4/4 system size with heavy work, but the shape of the metrics curve is representative for all scenarios and system sizes.

The figure shows that the MOS metric is the first metric that crosses the limit. System administrators that monitor the system to detect and mitigate any performance issues want to have an early warning on system saturation. For this purpose, the MOS value is the best candidate to be monitored. The other metrics could also be monitored, but they would give a later warning.

Figure 4 does not include the performance counters from the LyncPerfTool clients. These counters were always 0 when the highest possible number of users were reached. Small deviations of up to 0.04 SIP error responses/sec were

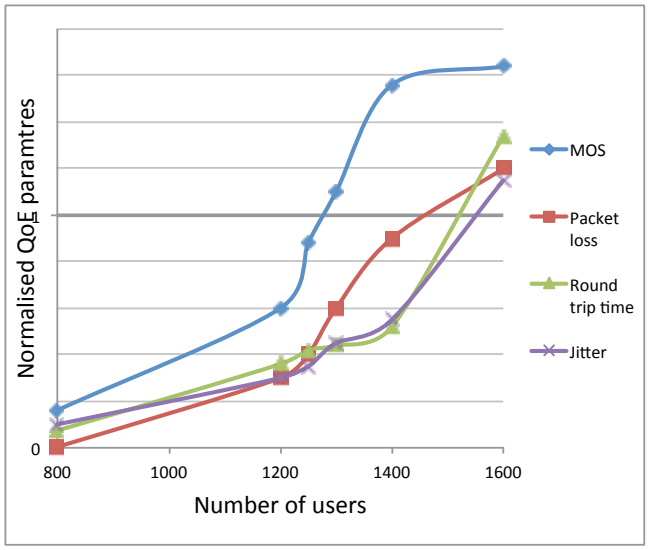


**Figure 4: Normalised QoE parameters, for 4/4 system size with heavy work.**

observed when the system was highly over-utilised. The SIP error messages are not considered a good metric for monitoring system saturation. The monitoring server metrics consider the real-time performance of ongoing voice sessions, etc. Real-time traffic is especially vulnerable to system contention and this may be the reason why the quality metrics gave an earlier warning compared to SIP error messages.

#### 5.3.5 *Find Capacity per System Size*

In our case, this is already done since we know that MOS degradation is the limiting system quality, and since we then have this information from Table 4 and Table 5. We will analyse the resulting scalability more closely in Section 5.4.

### 5.4 Analyse Results

We are now in a position to analyse the results of the tests. We will first find the bottleneck resource and will then explore the scalability before we finally try to use the results for extrapolation.

#### 5.4.1 *Find Bottlenecks*

Four resource types were considered in this study. Which of them was the bottleneck resource?

**Memory** All servers were provisioned with memory resources according to Microsoft's server recommendations. If any of the servers had too little memory, they would need to start swapping memory to disk, but memory swapping was always 0. Therefore, memory was not a bottleneck.

**Network** All servers had access to a 10.0 Gbit/s network interface. However, only the mediation server, edge server, and reverse proxy server used this interface. The highest network utilisation of all servers combined where 267 Mbit/s (for light work on the 4/4 system size). Not including hypervisor overhead, this gives a total utilisation of 2.67%, far below the utilisation of the processors. Network resources were therefore not a bottleneck in this Lync deployment.

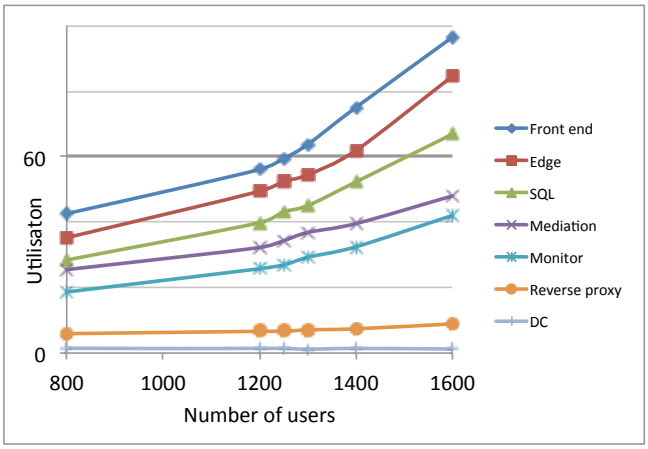


**Figure 5: Normalised utilisation of all servers, for 4/4 system size with heavy work.**

**Disk** The disk utilisation was evaluated by observing the time required to complete disk operations. As long as this time is kept below 15 ms, there is no contention on disk resources [18]. The highest observed disk service demand was 5.903 ms. This indicates that there was not a high enough utilisation of disk resources to create contention, and as a result disk resources were not a bottleneck.

**CPU** The utilisation of CPU resources closely correlates to system performance. Table 4 and Table 5 show that the utilisation of the front end server ranges between $58 - 60\%$ for all maximum users measures. The processor is the highest utilised and should therefore be considered the bottleneck in this Lync deployment.

A closer look at the utilisation of each VM may give more insight into how the system behaves under different CPU loads. Figure 5 shows the normalised CPU utilisation values for all servers in the 4/4 system size with heavy work. From the figure, it is possible to observe a breakpoint in the front end server utilisation curve when the maximum number of users is met. Similar break points are also observable in other servers, such as the edge server. However, on the edge server, the break point comes after the maximum number of users is reached.

Looking at Table 4 and Table 5, we see that the utilisation of the front end server is always below 60% for all measurements where the quality metrics are within limits. For all other measurements, the utilisation is above 60%. It is therefore reasonable to say that monitoring the utilisation of the front end server would be a good indicator of system health. This measure could be used by system administrators. They could even set an alert at 50% utilisation just to get an early warning on possible performance issues and therefore have the time to mitigate the consequences.

#### 5.4.2 *Explore Scalability*

Summarising the results from Table 4 and Table 5, gives two scalability curves for heavy and light work shown in Figure 6. From the table, we extract and plot the samples which approach the MOS degradation limit of 0.5 as close as possible without exceeding it.

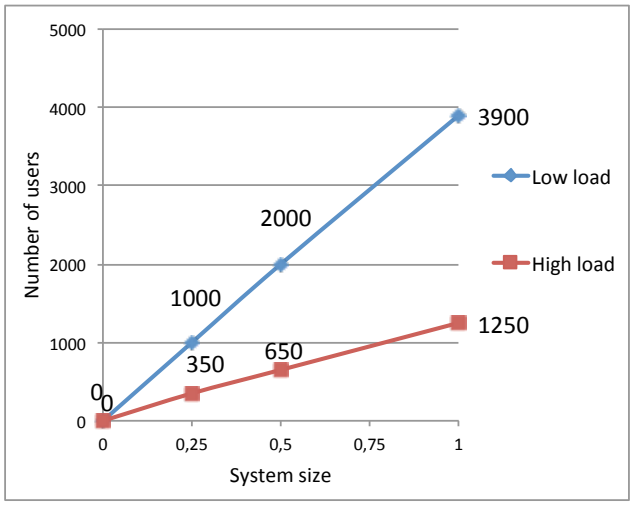


**Figure 6: Measured scalability in Microsoft Lync**

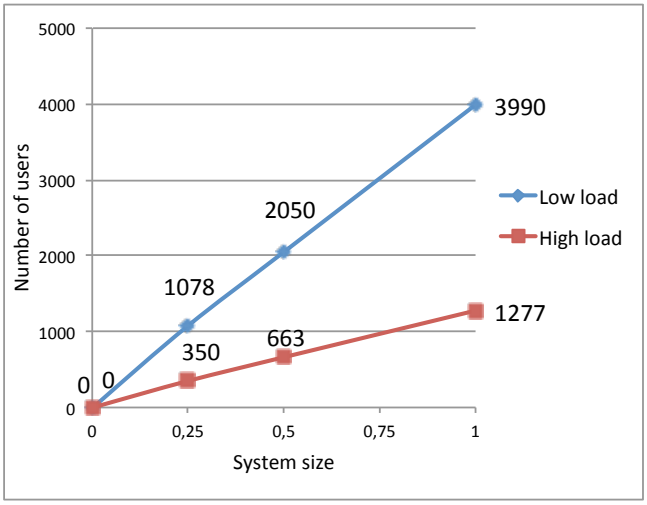


**Figure 7: Interpolated scalability values**

The results show a close-to-linear scalability. There are, however, some inaccuracies when measuring the maximum number of users:

- The binary search stops at a given number of iterations. This means that the unexplored search space may still be significant. Finding the "correct" value would require even more iterations, but was not feasible because of time constraints.
- Random measurement errors affect each test. This effect could be reduced by performing the same experiment several times and averaging the results. This would, however, require more time.
- Only the CPU resources are scaled, while memory, network, and storage remain the same for all system sizes. Even though the CPU has proved to be the bottleneck device, the improper scaling of other resources could create a bias towards higher number of users for 1/4 and 2/4 system size.
- The L3 cache in the 1/4 system size is scaled to 1/2 instead of 1/4. This may give a bias towards higher number of users because the processor can serve a relatively higher workload before experiencing contention because of the increased cache size. This is especially relevant since the processor is shown to be the bottleneck device of this Lync deployment.

The first inaccuracy in the above list can be reduced by using interpolation techniques. Section 4 describes why a binary search is preferred before an interpolation method when finding the maximum number of users. The quality parameters do not guarantee linearity. It is, however, safer to assume linearity over a smaller range, like the remaining search space after using the binary search for several iterations. This technique gives a more precise estimate for the maximum number of users. However, it is important to note that the MOS score is given in two digits only, so the round off error must be considered. The interpolation technique for finding the interpolated number of users ($N_{0.5}$) as well as lower and upper round off counters ($N^-_{0.5}$, $N^+_{0.5}$) is shown below

$$N_{0.5} = N_b + (N_a - N_b)\frac{(0.5 - MOS_b)}{(MOS_a - MOS_b)}$$
$$N^-_{0.5} = N_b + (N_a - N_b)\frac{(0.5 - MOS_b - 0.005)}{(MOS_a - MOS_b + 0.01)}$$
$$N^+_{0.5} = N_b + (N_a - N_b)\frac{(0.5 - MOS_b + 0.005)}{(MOS_a - MOS_b - 0.01)}$$

$N_b$ and $MOS_b$ denote the values measured *below* the MOS limit. $N_a$ and $MOS_a$ are *above* the limit. $N_{0.5}$ denotes the maximum number of users when the MOS value is interpolated to the limit value of 0.5.

The interpolation results are summarised in Table 6 and also shown in Figure 7. The figure shows the $N_{0.5}$ values only. The error bars ($N^-_{0.5}$ and $N^+_{0.5}$) are not included in the figure because they were too small to view.

The interpolated values for number of users still show close-to-linear scalability. There is, however, a slightly sub-linear tendency in the results. This tendency still exists when the round off error is considered. The improper scaling of resources (as explained in point 3 and 4 in Section5.4.2) should both give a systematic bias towards higher number of users for both 1/4 and 2/4 system size. However, it is hard to determine whether this bias alone accounts for the sub-linearity or whether the system really has a sub-linear scaling. Answering this question would require a different test setup where the bias was reduced or removed. An alternative test setup would be to run multiple identical Lync in-

**Table 6: Scalability results with linear interpolation**

| **Work** | **Heavy** | | | **Light** | | |
|---|---|---|---|---|---|---|
| **Size** | **1/4** | **2/4** | **4/4** | **1/4** | **2/4** | **4/4** |
| $N_b$ | 350 | 650 | 1250 | 1000 | 2000 | 3900 |
| $N_a$ | 400 | 700 | 1300 | 1100 | 2200 | 4050 |
| $MOS_b$ | 0.5 | 0.45 | 0.44 | 0.43 | 0.47 | 0.47 |
| $MOS_a$ | 0.59 | 0.64 | 0.55 | 0.52 | 0.59 | 0.52 |
| $N^-_{0.5}$ | 348 | 661 | 1273 | 1065 | 2038 | 3963 |
| $N_{0.5}$ | 350 | 663 | 1277 | 1078 | 2050 | 3990 |
| $N^+_{0.5}$ | 353 | 665 | 1283 | 1094 | 2064 | 4031 |

stances on the ESXi host instead of trying to scale down the system resources. This means that for the 1/4 system size scenario, a total of 4 identical Lync instances with the same workload would run at the same time. Then the available resources for each Lync instance would be 1/4. However, there would be more contention on resources. This kind of test setup with multiple Lync instances could not be tested because installing Lync instances was outsourced from Telenor. New Lync instances would have to be ordered and the resulting delay would postpone the master thesis work.

For practical concerns, such as when new Lync deployments are provisioned, the scalability in the 1/4 - 4/4 system size range can be considered linear. The sub-linear tendency that may exist in this system size range is too low to make an impact. This does not, however, prove that there are no scalability problems in Lync. It only shows that we have not found it in our system size range.

As shown in Table 1, the 1/4 system size was not correctly scaled. The amount of network, memory, and storage resources are not scaled at all. In addition, the level 3 cache is the same as in the 2/4 system size. A larger L3 cache will allow more instructions and data to be cached and increases performance on CPU intensive operations. As explained in Section 5.2.3, there are actually three MOS values, where the conference MOS value is highest in most scenarios, but not for some measurements for the 1/4 system size. Conferencing is a CPU intensive task (with considerable audio codec processing, in contrast to packet forwarding for peer-to-peer calls and PSTN calls). However, when the L3 cache is twice as big (compared to a "correct" scaling) the audio processing is better and other MOS values becomes the dominating factor.

### 5.4.3 *Extrapolation Beyond Size Range?*

Scalability testing can only be reliable within the system size range that was actually measured. In this paper the range is 1/4 - 4/4 physical servers. Using the Cisco UCS platform makes it quite straightforward to increase the number of blade servers and include them in the same virtual resource pool. For example, this would allow the system size to scale to 16/4 by including 4 blade servers. However, our standard edition license does not allow this.

To set up several servers in a pool, an enterprise edition license is needed. With the given constraint, the best configuration to achieve high performance is to put the front end server VM on a single physical server and allocate 24 vCPUs for it. Then the remaining VMs can be put on the other blade servers. However, the front end server VM requires more compute resources than the other VMs all together. Therefore, using more than two physical servers would not give significantly better performance, because the front end VM would have a much higher utilisation and become a bottleneck.

The Lync sizing study [18] shows that Lync scales linearly when the system size is varied from one to four physical front end servers. This sizing study uses the enterprise edition license and has a different user scenario than this report. However, the linear scalability conclusion from the sizing study supports the belief that the sub-linear scaling tendency observed in this report is actually caused by measurement bias alone, as discussed in section 5.4.2. Also the edge servers can be replicated on several servers.

**Table 7: Approx. time per binary search iteration**

| Time | Task |
|---|---|
| 5 min | Create user profile configuration file |
| 5 min | Prepare parameter capture scripts on hosts |
| 5 min | Copy configuration file to all 3 LyncPerfTool clients and start the LyncPerfTool clients |
| 90 min | Run test (30 min transient + 60 min meas.) |
| 10 min | Copy csv log file from hosts to home computer |
| 5 min | Read QoE parameters from the monitoring server |
| 10 min | Parse the results from the csv file |
| 5 min | Compare parameters to limits and determine the correct direction for the next iteration of the binary search |

## 5.5 Human Time Consumption

Considerable human time was required to set up the test system, tune the Lync configuration to work with LyncPerfTool, resolve issues with the performance measure tools, etc. This work would require less time for a person experienced with the specific technology who had direct access to the equipment instead of working remotely (as in our case). However, the time required to perform the actual tests would not have been affected by such concerns. The binary searches in this report were carried out in a total of 12 days. The work required for one iteration of the binary search is outlined in Table 7.

Table 7 shows that one iteration can be carried out in $\approx$ 2 hours 15 min. Configuring the tests and interpreting the results takes $\approx$ 45 min, half the time of actually running the test. Of course, this requires all steps to be carried out correctly. Since LyncPerfTool clients were situated both outside and inside the firewall (see Figure 3), an extra network interface was added to give external user simulators direct access to the internal network. This made file sharing possible without altering the standard reverse proxy configuration. This interface was for pre-test file sharing only. If the extra network interface was not disabled before the test was started, all sessions that were set up using ICE would short-cut the edge server and use the extra network interface instead. Then the performance results were biased towards better performance estimates and the whole iteration had to be run all over again with a disabled network interface. This happened twice during the test period and affected the total test time.

Automating the parameter capture could save a lot of manual work. It is possible to write scripts to start LyncPerfTool and to automate almost all parts of the capture process. However, there is one major and two minor issues that impede full automation. The major issue is that csv log files had to be copied from ESXi hosts using a Citrix Reciever GUI. Seemingly, there were no CLI options available. The minor issues were (1) creating new configuration files from UserProfileGenerator without using the GUI and (2) reading monitoring server results from a web interface. These minor issues could probably be overcome.

A person with experience in Lync setup can configure a Lync environment in $2-3$ days. An additional $2-3$ more days are required to install the LyncPerfTool and configure the Lync scenario accordingly. Applying the same method as described in this report, the results can be reproduced

in 16 – 18 days. If a different management setup were used (excluding remote access through Citrix Receiver), the parameter capture using the binary search could probably be automated.

In our tests, we have employed binary searches to find the optimal QoE for each system size and load. Because we now know that the scalability was almost linear, we could have started this binary search algorithm with more optimal values, and not exploring very low or very high values for each system size and load. The saved time could have been used to get a better knowledge of the confidence intervals for the most important measurements.

# 6. CONCLUSION AND FURTHER WORK

This paper has studied the scalability of Microsoft Lync 2010 in a virtualised lab environment. The results could help to optimise the provisioning of computer resources. The lab servers where deployed on a Cisco UCS platform with one blade server. We have used both light and heavy work on the Lync Server 2010 Stress and Performance Tool. The results show that there is a close-to-linear scaling from 1/4 to 4/4 system size. A new deployment should be scaled according to the number of simultaneous end users.

When the Lync deployment is installed on a Cisco UCS platform, the CPU resources become the bottleneck. The hardware configuration of Cisco UCS is comparable to hardware from other vendors. This means that upgrading CPU resources is most important for increasing system capacity.

Among several QoE (Quality of Experience) metrics, mean opinion score (MOS) is the best metric to monitor reduction of service quality for end users. The results shows that this metric is the first to exceed limits as the system load increases. Another good performance measure is to monitor the utilisation on the front end server. When this utilisation exceeds 60%, service quality is degraded.

From the findings in this paper, there are several interesting problems that need further study:

- Test Lync with more heterogeneous work than what was possible with the LyncPerfTool, to make the conclusions more robust.
- Explore the scalability of Microsoft Lync using an enterprise edition deployment which allows several servers in the same server pool. This would allow more front end and edge servers (scaling out), instead of just improving one front end and edge server (scaling up), as shown in this paper.
- We need experience with other scalability testing scenarios to further improve and validate the method.
- The utilisation of this scalability method for optimal provisioning should be described more comprehensively.

# 7. ACKNOWLEDGEMENTS

This research was supported by Telenor ASA. We thank Georg Lothe and Alexander Botnen, both in Telenor Norway, for solving many problems during the measurements.

# 8. REFERENCES

[1] Interpreting esxtop. `http://communities.vmware.com/docs/DOC-9279`, Dec 2008. Last visited: 2013 Feb 5th.

[2] Mean opinion scores and metrics. `http://technet.microsoft.com/en-us/library/bb894481(v=office.12).aspx`, Jul 2010. Last visited: 2012 Jul 8th.

[3] Lync stress and performance tool. `http://www.microsoft.com/download/en/details.aspx?id=25005`, Mar 2011. Last visited: 2012 Jul 8th.

[4] Media quality summary report. `http://technet.microsoft.com/en-us/library/gg615012`, Jul 2011. Last visited: 2012 Jul 8th.

[5] Unified computing technology. `http://www.cisco.com/en/US/products/ps10265/technology.html`, Feb 2012. Last visited: 2012 Jul 8th.

[6] L. A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12):33 – 37, 2007.

[7] G. Brataas, P. H. Hughes, J.-A. Fagerli, and O. C. Landmark. Exploring Architectural Scalability. *Software Engineering Notes*, 29(1):125 – 129, 2004.

[8] L. Duboc, D. Rosenblum, and T. Wicks. A Framework for Characterisation and Analysis of Software System Scalability. In *European Software Engineering Conference*, pages 375 – 384. ACM, 2007.

[9] N. Huber and et al. A Method for Experimental Analysis and Modeling of Virtualization Performance Overhead. In *Cloud Computing and Services Science*, pages 353 – 370. Springer, 2012.

[10] International Telecommunication Union. Methods for Subjective Determination of Transmission Quality. Aug 1996.

[11] P. Lewis, Abbate. *Microsoft Lync Server 2010 Unleashed*. SAMS, 1st edition, 2011.

[12] Microsoft, `http://www.microsoft.com/download/en/confirmation.aspx?id=6412`. *Quality of Experience*, Oct 2007. Last visited: 2012 Jul 8th.

[13] Microsoft, `http://www.microsoft.com/download/en/details.aspx?id=21646`. *Microsoft Lync Server 2010 Planning Guide*, Aug 2011. Last visited: 2012 Jul 8th.

[14] Microsoft, `http://technet.microsoft.com/en-us/library/gg398835`. *Server Hardware Platforms*, Mar 2011. Last visited: 2012 Jul 8th.

[15] Microsoft, `http://www.microsoft.com/download/en/details.aspx?id=12295`. *Microsoft Lync Server 2010 Capacity Calculator*, Feb 2012. Last visited: 2012 Jul 8th.

[16] K. H. Rygg. Scalability Modelling for Optimal Provisioning of Data Centres in Telenor. Master's thesis, NTNU, Trondheim, Norway, July 2012.

[17] P. Seeling. Network Performance Evaluation of Microsoft Office Communications Server 2007. In *IEEE International conference on Electro/Information Technology (EIT), 2011*, pages 1–6, May 2011.

[18] G. E. Solutions. Sizing Study: Virtualized Microsoft Lync Server 2010 Deployment. Apr 2012.

[19] VMware, `http://www.vmware.com/files/pdf/perf-vsphere-cpu\_scheduler.pdf`. *VMware vSphere 4: The CPU Scheduler in VMware ESX 4*, Des 2010. Last visited: 2012 Sept 20th.

[20] H. Winters. *Mastering Microsoft Lync Server 2010*. Sybex, 1st edition, 2012.