White Paper **INTERNAL**

Document Version: 1.0 – 2016-06-01

Using Single user Tests or Load Tests for Sizing?

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressly prohibited, as is any decompilation of these components.

Documentation in the SAP Service Marketplace

You can find this documentation at the following address: http://service.sap.com/sizing.

Document History

Version	Date	Change
1.0	<2016-06-01>	Version 1.0

Table of Contents

1 Introduction

Usually, when facing the need to evaluate what hardware would be required for target number of users, load testing come as first idea.

The effort starts with finding appropriate hardware for the load tests environment.

The typical "chicken or the egg" dilemma sounds like

- If I would have known the resource requirements of the application in advance, I would have known what hardware to buy and how to optimally setup the components on the different machines for maximizing the throughput.
- Because I do not know the resource requirements of the application in advance, I need to install and configure landscape somehow, to be able to understand what the resource requirements of the application are.

As a rule the system should have significant CPU resources, because on smaller servers some locking situations, which turn out to be scalability bottlenecks, may be overseen.

Example

A lock which takes 100-200 milliseconds would not be recognized on system with 4 cores, because the 4 cores could be allocated by concurrent users. But on system with 16 cores for example, the response time would grow with increase of the concurrent users far before full CPU allocation is achieved. Therefore the same product which is evaluated as scalable on the 4 cores hardware will be evaluated as non-scalable on the 16 cores hardware.

Same applies for the RAM of the server, the speed of disk I/O and disk space and the requirements to network capacity and latency between the different components. To guarantee successful load testing, significant amount of memory needs to be planned (e.g. to avoid swapping) or bottlenecks on disk and network I/O.

This means that significant money investment for buying or renting big server would be necessary.

This paper describes an approach for determining the hardware resource requirements of the application without investment in special hardware and with no need to run a load test.

2 Sizing, based on Single User Measurements

The resource consumption of the application can be determined already at the time it is developed, reusing the hardware of the development or functional testing system.

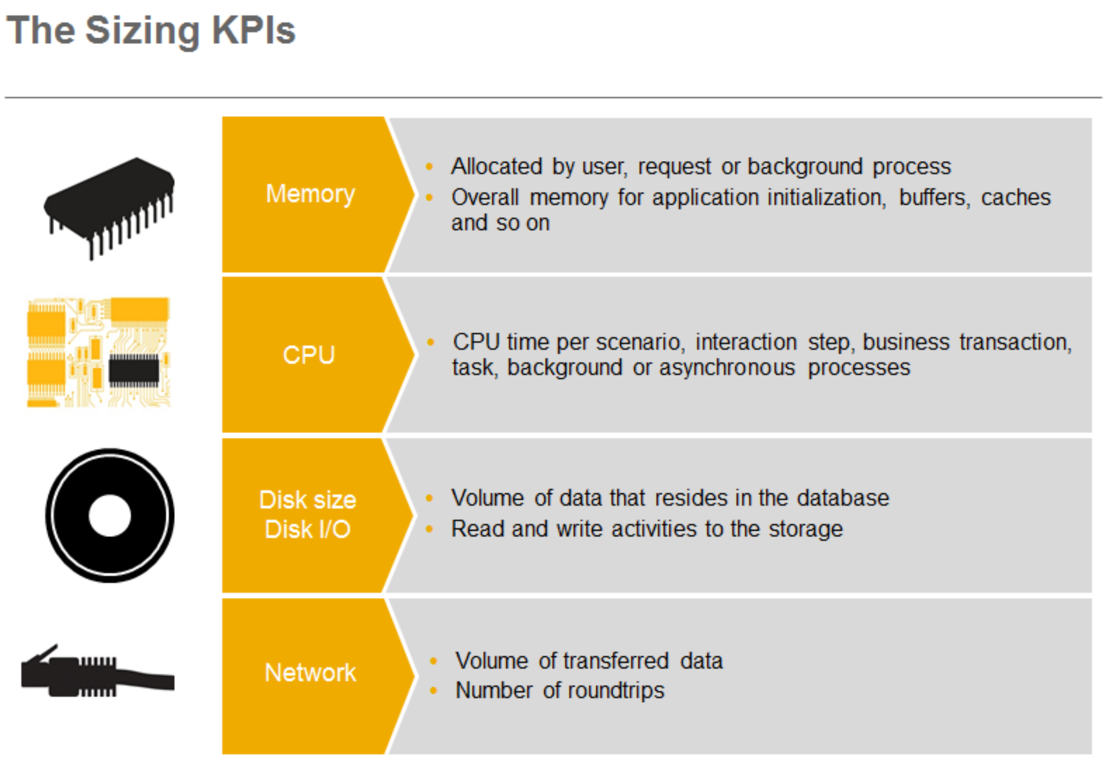
SAP has developed methodology which **based on single user performance KPIs** creates **sizing (extrapolation) formulas**, applicable to predict resource consumption of any target throughput and amount of users.

Even more - the methodology is flexible enough to handle variations in the business scenario flows and differences in object sizes by allowing **as many as necessary input parameters to the sizing (extrapolation) formulas**.

This sizing methodology exists since more than 20 years and is successfully applied to all SAP products. All official sizing guidelines provided by SAP at https://service.sap.com/sizing (in Quick Sizer, or as Sizing Guideline documents) are prepared following this single user measurements approach.

Each hardware resource (CPU, memory, hard disk and network) is associated with a lower or higher initial and maintenance cost and is part of the customers' total cost of ownership (TCO). This is why the prediction of hardware usage for target load, i.e. the **hardware sizing**, is so important for the SAP Customers.

There is a defined set of **Sizing KPIs** that should be measured and evaluated to fully understand how the application is using the hardware resources.



The Sizing KPIs

Memory
- Allocated by user, request or background process
- Overall memory for application initialization, buffers, caches and so on

CPU
- CPU time per scenario, interaction step, business transaction, task, background or asynchronous processes

Disk size
Disk I/O
- Volume of data that resides in the database
- Read and write activities to the storage

Network
- Volume of transferred data
- Number of roundtrips

The **CPU time** of business transactions or tasks is measured to determine the required number of Processors (Cores/Threads). It should be measured with appropriate granularity – for example, it is more useful to know the CPU time for each dialog step of the business process and not just for the complete business process.

If CPU is measured for each dialog step, then calculation of resource consumption for variations of the business process (including/excluding dialog steps) could be successfully made. The same is valid for the other hardware resources.

The **disk size** and **disk I/O** are measured to determine the rate of database table growth and the file system footprint. To determine the disk space growth the number of insert operations for the database, write operations for the file system, and bytes that are written to the database or file system are measured.

The total I/O rate calculation is based on measurements for all database and file access operations.

The **memory** is measured in different ways depending on the type of application. For some applications, it is sufficient to measure the user session context only, while for others it is also necessary to measure the application-specific buffers and shared spaces, temporary allocations by stateless requests, and so on.

The **network** measurements refer to the number of roundtrips for each dialog step and the number of bytes sent and received for each roundtrip. The measurements are used to determine the network bandwidth requirements for the target throughput and give recommendation on the network latency.

For example, applications that have high number of roundtrips per dialog step and have defined ambitious response time requirement may not be able to fulfil the response time requirements even in LAN.

The feasibility of taking reliable measurement for the above defined Sizing KPIs depends on the platform which is used by the application.

Most of the SAP Platforms - SAP NW ABAP Server, SAP NW Java Server, SAP HANA, SAP HCP provide performance measurements services (e.g. Performance Statistical Records, Communication Layer Traces, etc.) for all required Sizing KPIs.

For applications that do not run on the listed above platforms, but run directly or indirectly on top of SAP JVM, the Sizing KPIs can be measured with single user too.

Applications that are neither ABAP-based nor Java-based or HANA-based but run as standalone processes on OS level can be measured with single user with limitations. The limitations come from the available OS level performance measurement tools and traces. Anyway, the limitations of those measurements are less or same as the limitations of measurements with load tests, where same tools are used.

Details on how to perform measurements of sizing KPIs and how to prepare the sizing extrapolation formulas are provided in the standard SAP development trainings ZDS406 and ZDS415, as well as in customer workshop ADM115.

For a shorter introduction an e-book is
http://service.sap.com/~sapidb/011000358700000461812012E/index.htm available too.

3 Deficits of Load Testing for the purpose of Sizing

Few examples of problems which may occur in load testing and do not occur in single user testing may support the statement that load tests are not the right choice for creation of sizing guideline.

3.1 Oversizing

Background jobs, activated periodically, which are not part of the application but run in the same OS process, consume resources. This background noise could be reason for significant oversizing of hardware resources.

Load tests

With load tests resource consumption which is not function of the generated application load cannot be isolated, because during a load test the resource consumption is measured for the complete OS process (e.g. CPU per OS process, memory per OS process, I/O per OS process) or even total for all processes.

This means that if based on resource consumption monitoring for scenario with, for example, 100 requests per second, a conclusion about required resources for 200 requests per second would be made, the background jobs resource consumption would be counted twice, even when the background job would still consume same resources as with 100 requests per second. In some cases this can result into significant hardware oversizing (extra cost).

Single User tests

With single user tests methodology only the resource consumption for the application functionality is measured and the problem with the overhead of non-application specific resource consumption does not exist when extrapolating for higher throughput and number of users.

3.2 Scenario variants

Load tests

The load generation scripts follow strictly defined flow of scenario steps. The resource consumption, monitored during execution of this load test, is valid only for the concrete scenario scripts with concrete user's "think time" and concrete input parameters, or in more complex load testing projects - valid for the mixture of scripts and ranges of parameters.

In reality there may be multiple variants of the business scenario flow which the application user would perform. For example, for some online shops it may be specific that the "Buy a product" scenario includes 1 product search only before add to basket, while for other online shops the " buy a product scenario" may include 10 product searches before add to basket, etc. It is high effort to create load generation script and measure all variants of number of searches for products in the "buy a product scenario". The "Buy a product" scenario does not include only the search of product but many more steps, so basically due to the context of the user the searches cannot be isolated in separate load testing script. Even when they could, this example demonstrated that it is high extra effort to evaluate the impact of a variance in the user behavior on the resource consumption requirements if a separate load test should be performed for each scenario variation.

Single User tests

With the single user tests methodology the resource consumption is measured for each scenario step and therefore it is very easy to re-calculate resource consumption for a variant of a scenario flow.

In the concrete example, the resource consumption with 10 searches will be calculated by adding resources for extra 9 searches to the resource consumption of scenario with 1 search. The extrapolation formulas will allow flexibility on the number of searches by offering an input parameter for the expected number of product searches in one scenario, and in this way they will be able to calculate resources for scenario with any number of searches.

3.3 Measurements

Load tests

Most of the OS Platforms provide extensive monitoring of resources by activation of different OS commands. Examples of typically used monitoring tools on Linux are: *ps*, *netstat*, *top*, etc.

A single tool that provides all sizing information and for getting complete overview the output of different tools need to be collected with sufficient granularity, parsed and aggregated to common format and correlated to the load generator statistics, is not available. For this task, in practice, the testing team always spend time to write own scripts for parsing and correlating the outputs of the different OS tools.

The practice shows that even at expert level of competences in load testing, at the end not all information collected and correlated is really useful and some information is always missing.

Frequently there are doubts of how to use the collected data - for example in OS monitoring the CPU usage is provided in % and at the end it needs to be converted into seconds per user, so that it can be applied to predict CPU allocation on machine with different number of Cores. How to convert CPU % usage into CPU time is very common discussion topic.

Single User tests

The SAP tools for single user measurement collect the Sizing KPIs with no additional effort, i.e. for the official SAP Platforms the performance statistical records are always activated. There are SAP standalone measurement tools, which filter the results of the official statistical records and aggregate the sizing KPI values per business process dialog step - from this state on the preparation of sizing formulas is only mathematics.

The single user measurement tools provide the sizing KPIs already in the required metric, e.g. CPU time and not CPU% allocation and allow easy conversion to the hardware independent unit SAPS.

3.4 Side effects

Side effects while load testing can either make the results significantly better than they will be in reality or significantly worse.

Example

Database content (as test data), which is not well randomized to reflect real production volumes could result into locking of records that are touched by all concurrent users and artificial bottleneck on waiting for unlock of records will spoil the measurements. In reality, with productive data, such locking problems would not show up. With single user tests such problem will not be noticed too because only one user will lock the data.

Example

Insufficient resources at the load generator side, i.e. the load generator is not able to send sufficient number of requests to fully utilize the server while at the same time reports increased response times due to waiting times into its own queues can also be situation that puzzles the load testing team for a while. Usually multiple executions of the test (therefore wasted time) would be required to identify the reason for the non-scaling results.

With single user tests there is no resources bottleneck at client side in both cases - if executed manually or if executed in automated way.

Example

Many load generation tools do not have good representation capabilities for the result and provide only averaged numbers for response times per test run. With such a number only correlation of resource consumption to user load is very inaccurate, as it does not reflect any ramp-up, ramp-down, and peak or bottleneck situations. Deviations of the results are very hard to analyze.

With single user tests all sizing KPIs are mapped to the corresponding dialog steps of the scenario. The single user tests are executed multiple times and it can be clearly identified if KPIs values are "reproducible" (always in same range for each of the dialog steps). If the values are not reproducible, deep-dive performance analysis with profilers, traces and other development tools may be required only for the dialog steps which show the problem.

There are more examples.

4 The Good Goals of Load Testing

While the load tests are not a recommended option for sizing measurements, the load tests are a good option for

- verification of the sizing (extrapolation) formulas for concrete scenarios
- proving of vertical and horizontal scalability of software and landscape
- proving of stability
- evaluation of behavior in overload situations
- fine-tuning for maximum allocation of hardware resources

4.1 Sizing verification

When the sizing model, i.e. formulas with input parameters for the sizing relevant functionality is ready, the most frequently used or mission critical scenarios could be sized for defined target throughput.

The sizing result should be used for planning the hardware environment for a load test with appropriate configuration of the application server, database and other relevant components over the sufficient hardware.

The load test results for usage of CPU, memory, network and disk should match to the sizing results, for example the CPU usage should be in range of 65% as planned in sizing.

4.2 Scalability verification

The load tests are the methodology for verification of the scalability of the application and the landscape.

For verification of scalability, the load test should be run with progressive increasing generated load.

For proving the **vertical scalability**, the expected behavior is that the application throughput and resources allocation follows the increase of generated load in a linear way while response times remain constant for up to 65% of the resources allocation.

For proving of **horizontal scalability**, the throughput should increase linear to the additional hardware resources (with symmetrically installed software instances on it) if the generated load is sufficient.

4.3 Stability verification

The load tests are the methodology for verification of the stability of the application and the landscape. The stability tests should demonstrate that at constant load at 65% resource allocation the system can run for very long time (usually 24x7) without increase of the overall resources allocation, which means that there are no resource leakages.

4.4 Evaluation of overload situations

Combing the stability load test with temporary intervals of higher generated load to simulate peaks and overload the system is very useful as prove of stability too. During the **peak intervals** are expected resource usage of 100% and significant increase of the response times. Once the "overbooking" load simulation is stopped, it is expected that throughput and resource consumption fit again into the sizing expectation.

4.5 Fine-tuning

The sizing model guaranties **only** that the hardware resources would be sufficient for the target load. **The actual allocation of resources depends on the configuration of the software.**

Example

An ABAP server with default 40 work processes would not be able to fully allocate machine with 40 cores if there is database activity in each dialog step and the database is installed on different machine. Due to the time, spend in database I/O there should be configured more work processes, which can accept and process new requests to compensate the database response wait times and allocate the available CPUs better.

Example

In other technologies, for example Java, a sizing result of 32 GB memory with some applications may better fit to configuration with 2 Java Instances (each 16 GB heap) and with other applications (e.g. if they have huge spaces required for caches), may better fit to a configuration with 1 Java Instance (total 32 GB heap).

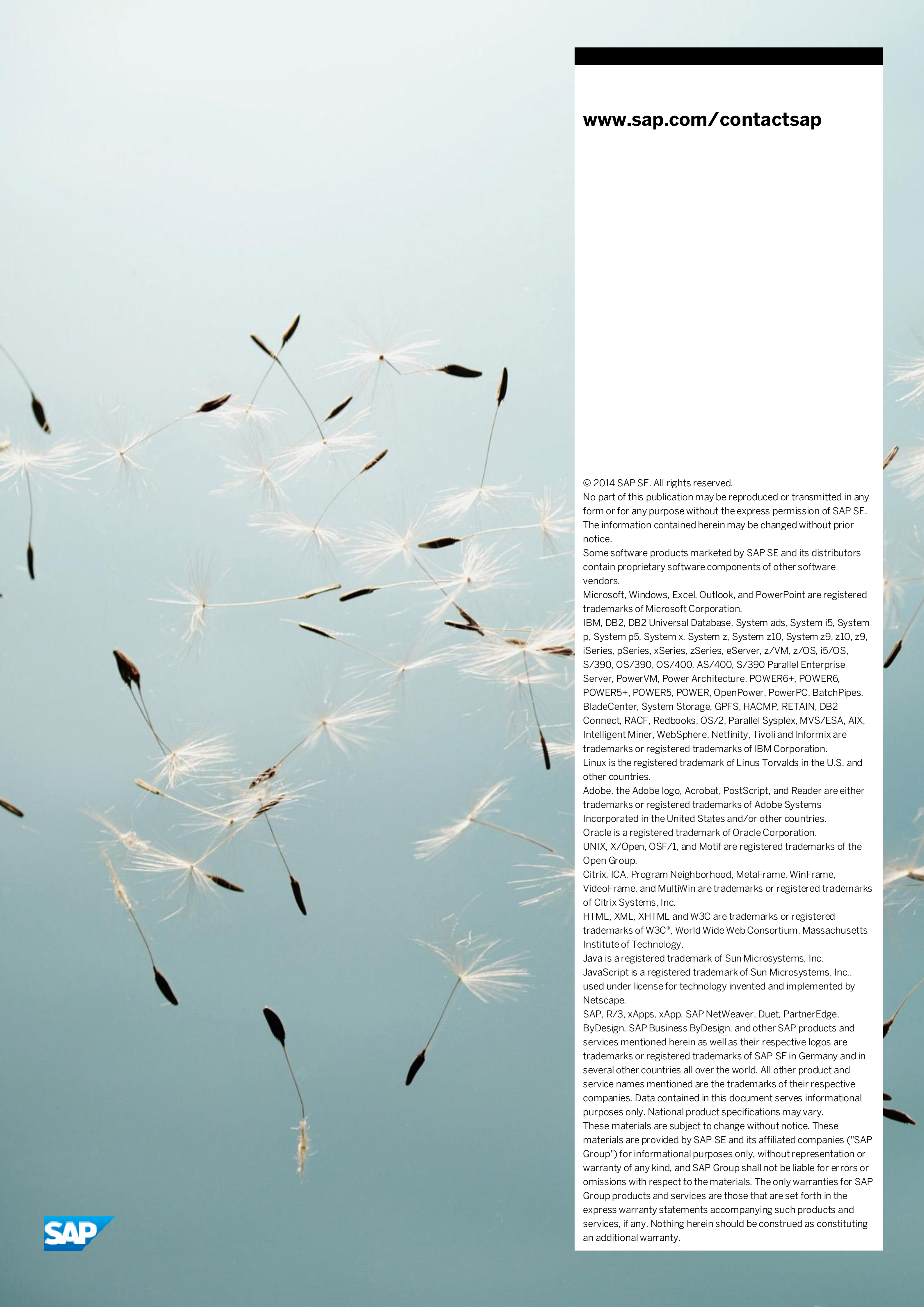
Load tests help to make sure that hardware resources shortage is not expected because the sizing model of the application, the configuration issues and tuning bottlenecks for a concrete application and scenario have been identified, resolved, validated with load tests and documented in tuning notes for customer.

5 Conclusion

Spare the time, the effort and the money and use the best approach for sizing, which is the creation of sizing (extrapolation) formulas, based on single user measurements.

Only for very high volume and mission critical scenarios, get confident about the scalability and stability qualities of the application by performing load tests on hardware environment that corresponds to the sizing result for the target load.

Analyze the relevant tuning and configuration options of the software to fully allocate the available hardware and document the best practices and tuning hints into relevant tuning notes or directly in the sizing guideline documents.



www.sap.com/contactsap