Volume test

Objective:

- **Find problems with max. amounts of data.**
- **System performance or usability often degrades when large amounts of data must be searched, ordered etc.**

Test Procedure:

- **The system is run with maximum amounts of data.**
- **Internal tables, databases, files, disks etc. are loaded with a maximum of data.**
- **Maximal length of external input.**
- **Important functions where data volume may lead to trouble.**

Result wanted:

- **No problems, no significant performance degradation, and no lost data.**

Considerations:

- **Data generation may need analysis of a usage profile and may not be trivial. (Same as in stress testing.)**
- **Copy of production data or random generation.**
- **Use data generation or extraction tools.**
- **Data variation is important!**
- **Memory fragmentation important!**

Volume test shall check if there are any problems when running the system under test with realistic amounts of data, or even maximum or more. Volume test is necessary, as ordinary function testing normally does not use large amounts of data, rather the opposite.

A special task is to check out real maximum amounts of data, which are possible in extreme situations, for example on days with extremely large amounts of processing to be done (new year, campaigns, tax deadlines, disasters, etc.) Typical problems are full or nearly full disks, databases, files, buffers, counters that may lead to overflow. Maximal data amounts in communications may also be a concern.

Part of the test is to run the system over a certain time with a lot of data. This is in order to check what happens to temporary buffers and to timeouts due to long times for access.

One variant of this test is using especially low volumes, such as empty databases or files, empty mails, no links etc. Some programs cannot handle this either.

One last variant is measuring how much space is needed by a program. This is important if a program is sharing resources with other ones. All programs taken together must not use more resources than available.

Examples:

Online system: Input fast, but not necessarily fastest possible, from different input channels. This is done for some time in order to check if temporary buffers tend to overflow or fill up, if execution time goes down. Use a blend of create, update, read and delete operations.

Database system: The database should be very large. Every object occurs with maximum number of instances. Batch jobs are run with large numbers of transactions, for example where something must be done for ALL objects in the database. Complex searches with sorting through many tables. Many or all objects linked to other objects, and to the maximum number of such objects. Large or largest possible numbers on sum fields.

File exchange: Especially long files. Maximal lengths. Lengths longer than typical maximum values in communication protocols (1, 2, 4, 8, … Mega- og Gigabytes). For example lengths that are not supported by mail protocols. Also especially MANY files, even in combination with large lengths. (1024, 2048 etc. files). Email with maximum number of attached files. Lengths of files that let input buffers overflow or trigger timeouts. Large lengths in general in order to tripper timeouts in communications.

Disk space: Try to fill disk space everywhere there a re disks. Check what happens if there is no more space left and even more data is fed into the system. Is there any kind of reserve like “overflow-buffers”? Are there any alarm signals, graceful degradation? Will there be reasonable warnings? Data loss? This can be tested by “tricks”, by making less space available and testing with smaller volumes.

File system: Maximal numbers of files for the file system and/or maximum lengths.

Internal memory: Minimum amount of memory available (installed). Open many programs at the same time, at least on the client platform.

General points to check

- Check error messages and warnings, if they come at all for volume problems and if they are helpful and understandable.
- Is data lost?
- Does the system slow down too much?
- Do timeouts happen? IN that case failures may also happen.
- If it looks like it goes fine, are really ALL data processed or stored? Even the end of files, objects or tables?
- Are data stored wrong?
- Are data lost or written over without warning?

How to find test conditions

Test conditions are data that could turn into a problem:

Find which input data and output data occur in an application or function.
Find restrictions for number of data, especially maximum and minimum. Include especially data, which are stored temporarily or are read from temporary store.

For every data element, check if there can occur larger volumes than allowed. Check what happens if data elements are counted. Can the maxima come out of bounds? What about sums, if many objects are summed up? Can that cause out of bounds values?

If a data element is stored temporarily, how much space is necessary? If there are many such elements, can the space be too little?

If there is any form of numbering or coding, are there restrictions in it that can preclude growth? For example if there are two character fields, there may not be more codes than 26*26 possibilities.

Find boundaries on system wide data, for example maximum disk volume, maximum number of files for the file system, maximal file lengths, buffer lengths etc. and look if any of them can turn into a problem.

Find restrictions in temporary storage media like maximal length of mail, CD, DVD, tape etc.

Find restrictions in communications, for example timeouts and maximal lengths of messages, files etc. Check how much data can be transferred before a timeout will come.

Find places where data is stored temporarily. Find the functions storing data there and functions reading and taking away these data. Make test scenarios (soap operas) going thorough both storing and deleting. Try to find out if the place can get full, and overflow problems. This requires long scenarios, maybe even random generated function calls. Check if an audit trail may lead to problems after logging many transactions.

Can volume test be left out?

The preconditions for taking away volume test is that volume questions are checked in earlier test and answered sufficiently well. This means volume is tested or checked in lower level tests or reviews and the results can be checked.

Caution when integrating several independent systems to be executed on the same platform: It must be guaranteed that every system has enough resources for itself. If one cannot guarantee that the platform delivers the necessary resources, in this case all kinds

of memory, then a volume test of all systems together should be executed with maximal data volumes for every system.

Checklist:

If at least one question is answered with NO, volume test is interesting.

- Is volume test executed before, on the whole system and can the result be checked?
- Can we guarantee that the system always has he necessary memory resources?
- Can we guarantee this if several systems share the hardware?
- Is it guaranteed that no larger data volumes than specified will occur?
- Is there a low risk if data volume turn greater than specified anyway but the system does not work well enough then?

Problems when generating data:

- Fragmentation of memory difficult to generate
- Relational integrity of generated data
- Dynamic generation of keys
- Data should follow the usage profile