Software Improvement Group

BEST PRACTISES

Ten best practices for a green IT system

RVO Energy Footprint Project, Knowledge Network Green Software (KNGS) and the Cluster Green Software

These best practices were developed within the RVO Energy Footprint Project, supported by the Knowledge Network Green Software (KNGS) and the Cluster Green Software. The Cluster Green Software project is a new technical and scientific regional cluster in the Amsterdam Metropolitan Area (AMA).

Within this cluster the participating organisations contribute towards work on:

- → The mapping of the energy consumption in (large) systems which is caused by the use of software and the search for opportunities to reduce the energy usage of software.
- → The development of tools for users of large software systems to enable them to manage energy costs.

Kngs.wikidot.com
www.clustergreensoftware.nl
www.sefindex.org

Professor dr. ir. Joost Visser
Chairman KNGS

1. Use a virtualised environment where possible

Footprint measurements clearly show that a virtual server is ten times more energy efficient than a physical server. The superfluous capacity of the server can be used by other applications. When creating the architecture of an application, bear in mind that all parts will be virtualised.

2. Make use of the energy efficiency settings offered by the hardware and virtualisation layer

Energy footprint measurements show that the use of energy-efficient hardware settings (for example, using CPU C-states) is not trivial in deployed servers and other hardware in the data centre. Even "pre-tuned variables" are changed to less energy-efficient settings without considering the consequences.

3. Provide a measurement infrastructure for determining energy KPIs during the rollout of the application in its production environment

Turning on energy measurements afterwards is not cost-effective and is often difficult. The availability of energy measurements is imperative for checking the energy efficiency of applications.

Software Improvement Group

4. Dare to consider approaches that lead to more energy effective solutions

The trends and effects of changes to the system can become apparent by using the measurement infrastructure. In many situations, the operations team is afraid to make changes and as a consequence energy inefficient situations persist. Lower energy efficiency comes to light if the system is experimented with and explored from the beginning, when knowledge about the system is still 'fresh'. Share the knowledge about the system among the teams, including the operations team.

5. Replace older hardware with new hardware in time

Older hardware (more than three years old) is less efficient than the latest hardware. The capacity and processing power of hardware (and software) improves every year. By postponing replacement, energy inefficient situations may continue to exist.

6. Limit the oversizing of systems

Projects and systems often need several years before the projected load on the systems is reached. During the first period of deployment, significant oversizing may occur. This is not a desirable situation, since the system is often not used in an efficient manner during this period. This situation can be prevented by not initially sizing the system according to its final dimensions; instead gradually increase its capacity. Redundant capacity should be released for other applications' use.

7. Reconsider availability requirements

Application owners tend to exaggerate the availability requirements to create safety margins for their application. Very often the requirements prove not to be realistic; however the infrastructure has already been delivered and is operational. In such cases it is necessary to have a process to reconsider the requirements. The software under investigation must support the possible outcomes (use less hardware/capacity).

8. Only activate the test and failover environment when needed

Configure the test and failover environment so that they are only switched on when they are actually needed. It is common for testing and disaster recovery environments to be left on continuously. Acceptance environments are only used during new release testing. Make starting and stopping a test and disaster recovery environment relatively simple.

9. Optimise for performance

Performance optimisations, like improving hardware resource consumption (CPU/memory etc.) relative to the amount of work (transactions), often lead to a reduction in energy consumption. A lot of tooling and experience is available for performance analysis, so use it! An environment is often sized based on its peak load. Increasing the capacity while using the same hardware can have a big effect on the energy-efficiency.

10. Take the workload pattern into account when sizing the system

Many systems show a pattern in the workload: a constant load or a peak once a day, week, month or year. Determine the anticipated workload and design the system in such a way that it can handle the variation. In a system with a constant load, less spare capacity is needed to cope with the variation. If a system shows an annual peak in its load, the system can be scaled down during the rest of the year.

This research was conducted in collaboration with:

Rijksdienst voor Ondernemend Nederland

Cluster Green Software

#KNGS
Knowledge Network Green Software
–Connecting the Green Software world–