A Green Software Development Life Cycle for Cloud Computing

Nitin Singh Chauhan and Ashutosh Saxena, *Infosys, India*

Cloud computing's recent proliferation has received attention from green crusaders hoping to mitigate the carbon footprint of large datacenters and IT infrastructures, but what about the software? A new framework for a greener cloud focuses on energy-efficient software development.

A common topic of deliberation is information and communication technology's effect on global warming and our environment. Many look to cloud computing as a solution because it lowers energy requirements by optimizing physical machines,[1] but this isn't necessarily true for all implementation scenarios.[2] We need to explore more opportunities to reduce carbon footprints in the cloud environment, evaluating the cloud's ability to save energy through both technology- and process-related changes. Here, we take a more holistic approach by putting forward a blueprint for a greener cloud methodology.

Organizations such as Greenpeace have been asking large datacenters to control their energy consumption and use clean energy resources.[3] However, you can't use the same approach to control energy consumption in the cloud as you would in a traditional data center, because the cloud presents a distinctive set of challenges. Prior work has focused on energy-consumption requirements and issues in traditional datacenter environments. [4] Researchers are also exploring how to optimize the underlying technologies of the cloud, such as virtualization,[5,6] and build energy-efficient hardware.[1] Researchers haven't paid as much attention to energy efficiency from the software perspective.

The typical software development life cycle (SDLC) has a vital role to play in building efficient software applications in a systematic way.

 Published by the IEEE Computer Society

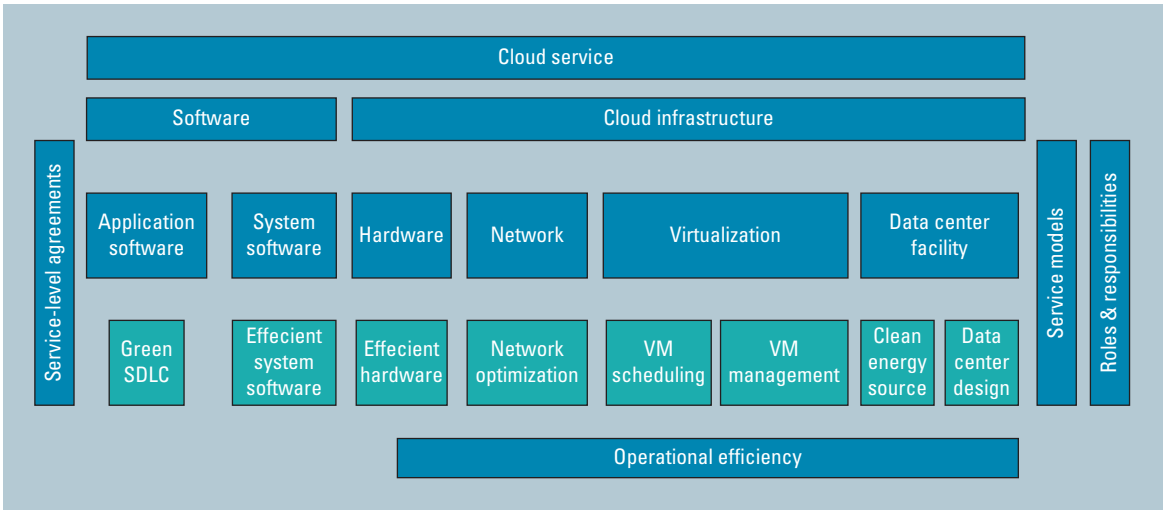

Figure 1. A green cloud computing framework. The framework is divided into two main components: the software and the cloud infrastructure. Blue-colored components in the figure indicate cloud services building blocks, and green indicates opportunities for energy efficiency against these components.

Here, we try to identify energy-saving opportunities in a typical SDLC process to help build more environment-friendly software applications for the cloud setup.

A Green Cloud Framework

We introduce an enhanced cloud framework that takes a holistic view of the cloud environment and maps energy-saving opportunities to various cloud components. A.J. Younge and his colleagues proposed a green cloud framework a few years ago, but it covered only virtualization and datacenter operations.[7] Our proposed framework also covers software development, which can significantly improve energy efficiency in cloud environments. As Figure 1 shows, we divide our framework into two major components: the software and the cloud infrastructure.

Service Level Agreements

Cloud customers expect a certain level of service from their cloud service provider (CSP). The customer and CSP usually agree upon the details in advance via a service level agreement (SLA). SLAs are stringent for applications that require high performance. They can affect the quality of service expected from cloud providers and influence the cloud's energy consumption. For example, if a mission-critical application requires high availability, the CSP must keep failover and backup resources running constantly to meet the SLA, negatively affecting energy consumption because some backup resources consume energy even when not in use.

Including energy-related terms and conditions in the SLAs is another way to track energy consumption in the cloud. Cloud customers and providers can outline SLA terms regarding energy-usage limits and responsibilities. Energy consumption beyond such limits should be recorded, and parties should be penalized for SLA violations. There can be mechanisms to award or transfer carbon credits to cloud stakeholders based on SLA adherence.

Software

Software is usually categorized as either application or system software. Users directly access application software to perform required business functions, while system software comprises low-level programs that hide the complexities of computer functioning from the user.

One component of the proposed framework is energy-efficient system software, which includes operating systems and compilers. The cloud environment is complex, and multiple platform types and underlying OSs enable its functionality. The OS performs the critical job of scheduling and allocating resources. Optimally scheduling algorithms and resource-allocation schemes can significantly improve energy efficiency. Furthermore, the OS design should help reduce the time required for computations. Support for multithreading in the OS can minimize the active CPU time by ensuring

work in parallel. Consequently, developers must consider energy-efficiency provisions while selecting the underlying system software and platform.

Generally, the cloud environment uses virtual machines (VMs) instead of dedicated physical machines. Developers should design VM software and OS instances using energy-efficient software development practices. They should also explore opportunities to customize OS elements, keeping only those necessary to execute the required task. Creating a lightweight VM instance can save energy by reducing the VM image boot time and storage.[7]

Cloud Infrastructure

The cloud infrastructure is another major constituent of our green cloud framework. It includes IT and facility infrastructure components in the CSP environment. CSP datacenters need various types of processing hardware, storage, and network devices to make the cloud operational. Furthermore, hosting such devices requires physical facilities, so the design and implementation of cloud-infrastructure components can significantly affect energy consumption.

The design, construction, and maintenance of facility buildings in datacenters greatly affect the environment and natural resources.

Hardware. The cloud IT infrastructure has many complex systems and a large pool of high-performance computing resources and high-capacity storage devices. Multiple cloud customers share these resources, which CSPs can provision as needed. Because scalability is a key cloud characteristic, CSPs must keep buffer resources to meet the customers' changing requirements and SLAs.

There are two issues with respect to cloud IT infrastructure. The first deals with having energy-efficient hardware and computing devices. The second focuses on optimally using hardware to minimize energy resources without compromising performance and security.

Network. The cloud leverages the Internet to deliver its services. In a traditional in-house datacenter, application building blocks are hosted in one centralized location. In the cloud scenario, network communication has increased significantly owing to the cloud's distributed nature and the mode of service offerings over the Web. This additional communication translates into additional energy consumption.[2]

Furthermore, service offerings over a public network have required additional security controls, which contribute to additional carbon expenses. An energy-aware framework needs network optimization to support a design that will minimize network communication.

Virtualization. Virtualization is a key enabling technology for cloud services. As we know, virtualization saves energy, because multiple logical machines can run on single physical box. However, we can further optimize this technology for increased energy efficiency.

VM allocation and scheduling schemes both have the opportunity to reduce energy expenses through efficient resource use. Many researchers have focused on VM allocation and management techniques in the cloud.[8,9] There are various ways to allocate VMs on a physical node, and VM distribution should minimize the number of servers running. Schemes can migrate VMs effectively from one box to another based on resource use without compromising the quality of service and performance as outlined in the SLAs. VMs from a low-load physical box can be migrated to other running boxes, and low-load machines can be shut down to save energy.[7]

Datacenter facility. CSP companies must pursue datacenter design strategies that can help reduce carbon footprints. Using dirty energy sources such as coal to meet CSP's datacenter energy requirements is a major concern. CSPs need to

- adopt clean and renewable energy source, such as solar, wind, ocean, and hydropower;
- develop cost-effective and environmentally friendly power-generation mechanisms; and
- store energy to better use energy sources and avoid waste.

The design, construction, and maintenance of facility buildings in datacenters greatly affect the environment and natural resources. Cloud facilities are huge, often consuming significant energy to meet lighting, cooling, and other operational requirements. Such facilities need green buildings that use environment-friendly materials and processes and that are resource-efficient throughout the life cycle.

Operational Efficiency

IT process improvements for cloud management services are another important dimension. Cloud stakeholders should reduce paper consumption and automate paper-intensive processes. Furthermore, supply-chain optimization, green procurement policies, efficient datacenter administration, and e-waste management are some other areas that can help reduce the carbon footprint.

Service Models, Roles, and Responsibilities

In the cloud environment, application development responsibilities depend on the adopted cloud model. In the case of software as a service (SaaS), the cloud provider is responsible for software development, while in platform as a service (PaaS), the cloud provider only provides the platform for developing an application. If the customer has subscribed to an infrastructure as a service (IaaS), all responsibilities of application development and maintenance lie with the cloud customer.

Another motive for building our framework was to highlight the different responsibilities of cloud stakeholders, who have different responsibilities, again depending on the adopted cloud model. CSPs should

- establish energy-efficient datacenter facilities and use clean energy sources;
- procure energy-efficient hardware and devices;
- optimize IT infrastructure design and implementation (in terms of the network, storage, servers, and so on);
- manage and monitor power;
- use energy-efficient operational controls;
- create an energy-efficient application development platform (in the case of PaaS);
- create an energy-efficient application (in the case of SaaS); and
- use energy-aware VM scheduling and management.

Cloud customers should

- ensure that the SLA includes energy- and performance-related terms;
- develop an energy-efficient IT infrastructure to interact with the cloud environment;
- develop a design that optimizes network communication; and
- accurately forecast resource requirements.

Cloud developers must follow an energy-aware SDLC process for developing cloud applications (for IaaS and PaaS).

Finally, users are also responsible for saving energy. They should close browser tabs and scripts when not in use, control data generation and use of social networking sites, and avoid wasting battery energy in mobile devices.

Introducing energy awareness early ensures that the energy-related expenses can be minimized in later SDLC phases.

The Green SDLC Principle

SDLC is a systematic process for developing software applications. Introducing any concept early in the SDLC can ensure that the concept is propagated throughout the various phases. For example, to build secure software, security concepts must be introduced in the initial SDLC phase. In the same way, introducing energy awareness early ensures that the energy-related expenses can be minimized in later SDLC phases.

Existing work highlights various approaches and guidelines for energy-aware software development,[10,11] but it doesn't cover the complete SDLC in the cloud context. Compared to traditional SDLC, energy-aware SDLC for cloud applications is influenced by two concepts—green computing and cloud computing. Here, we look at the five SDLC phases and discuss how green and cloud characteristics might affect them.

Software Requirement Specification

Typical system requirements are divided into two categories: functional and nonfunctional. Functional requirements relate more to expected business software functions. Nonfunctional requirements include security, flexibility, performance, reliability, robustness, availability, portability, and usability. A greener approach to software requirements specification should add sustainability to this list.

Relevant questions to ask during the requirements-gathering phase include the following:

- Are there any additional application requirements due to cloud characteristics—for example, additional security and communication requirements?
- Is there way to measure energy consumption by introducing an external API for the application in the cloud?

The cloud's Web service delivery model and the increasing popularity of smart gadgets highlight the importance of optimizing user interfaces for energy savings.

- Are there any benchmarks and metrics related to energy consumption for an application or for major transactions in an application?
- Is the environmental effect of application hosting in the cloud being considered during the feasibility study?

Design

Technically, the design phase could be separated into different classifications such as data, architectural, procedural, and interface design. However, most designs focus on modularity, abstractions levels, data structures, and software architectures.

A good design should also cover energy consumption. For example, highly dependent modules could lead to communication overhead in terms of parameters and control passing. Such a design would result in high energy consumption, which, in isolation, might not be noteworthy. However, in the case of repetitive communication over a certain time period, this could be significant. In a distributed model of cloud computing, such communication overhead could substantially increase energy consumption.

GUI design is another area where an effective design can save energy by minimizing user interactions.[12] An effective design should also use all available screen space, avoid vestiges of desktop systems, and cache frequently used input—especially when designing a user interface for mobile devices with limited battery power. The cloud's Web service delivery model and the increasing popularity of smart gadgets highlight the importance of optimizing user interfaces for energy savings.

Data structures and algorithms affect the efficiency of software applications and influence application performance and energy-use patterns. Unnecessary controls, branching, and loops in algorithms require additional computation resources. Designing an algorithm requires a considerable amount of analysis to reduce complexity. Also, deciding between existing algorithms requires considering the computational and storage complexities. For example, in a cloud scenario, encryption is a key functionality to address security-related challenges, and typically the AES (Advanced Encryption Standard) encryption algorithm consumes less energy than DES (Data Encryption Standard), 3 DES, or RC2 (Rivest Cipher 2).[13] An energy-efficient dataflow design for cloud applications should avoid redundancy and uncontrolled dataflows.

Implementation

Using energy-inefficient code in a traditional datacenter mainly harms the green cause, but when processing is based on a pay-per-use model, inefficient code also affects the enterprise's pocketbook. Therefore, writing efficient code in the cloud scenario can reduce both costs and energy consumption.

Modern applications use multiple application frameworks and have several tiers of abstraction that could distance the code from the processor. Such complexity might be useful from a programming perspective, but it doesn't favor the green cause. The programming language can affect an application's energy consumption.

Some programming languages can effectively use CPU and memory through techniques such as multithreading and garbage collection.

PaaS service providers offer a platform that includes a customized API to assist with cloud-enabled application development. This platform should be evaluated with respect to energy consumption as well, apart from development flexibility and capability.

Some established programming techniques help reduce an application's CPU consumption. The loop-unrolling mechanism reduces the instructions controlling the loop by rewriting them in a new sequence of instructions that can be executed in parallel. The declaration and scope of variables, variable types, switch statements, and nested loops are other programming techniques that can influence processing and energy consumption. All these energy-saving techniques also apply to cloud applications. In the cloud, applications might need to communicate and share parameters across the network. The number of parameters should be controlled to optimize energy consumption during information transmission.

Testing

A well-planned testing process can save time, money, and energy by identifying defects early. Testing itself consumes significant resources and is a critical SDLC phase. Tests related to sustainability should check energy consumption for major application functions and transaction cycles.

Test planning should consider

- the scope and objectives of testing energy consumption in the cloud;
- the approach for testing such consumption;
- the number of people and amount of equipment allocated for testing affect energy use; and
- how to measure energy use.

The testing team should create and execute well-defined test cases related to energy consumption. They should also analyze the results of security and performance testing to evaluate energy consumption trends in various conditions.

In the context of a cloud application, the testing process might differ based on the service model. Some application components might reside on the cloud, while others are in the enterprise environment. There could be additional flow of test data across the network during the testing process that increases the process's carbon footprint.

There are additional testing requirements with respect to network, performance, and security because of typical cloud characteristics. Because security and trust are major challenges in the cloud, security validation and quality assurance require additional effort. Furthermore, the cloud is a heterogeneous environment, and the complexity of testing and fixing bugs can result in additional effort, resources, and energy consumption.

Maintenance

Applications and the underlying infrastructures must be maintained effectively until system disposal. In the case of the cloud, CSPs control hardware and infrastructure monitoring, so they must track energy use.

Tests related to sustainability should check energy consumption for major application functions and transaction cycles.

Application and software maintenance responsibilities depend on the adopted cloud service model. In IaaS, the cloud customer is mainly responsible for the application, while in PaaS, the cloud provider should maintain the platform. In the SaaS model, accountability lies with the cloud provider for application maintenance. The software maintenance team of cloud providers and customers should ensure that released software patches and fixes don't negatively affect the application's carbon footprint.

Energy efficiency currently has a low priority for most cloud stakeholders, but increasing adoption of the cloud requires medium- and long-term green initiatives. Our proposed framework and energy-efficient SDLC

principles are guiding a manifesto in this area. Some of the framework's components should be addressed individually in the absence of a holistic green cloud approach. Owning and executing individual responsibilities discussed earlier will be a step toward energy-efficient cloud services in the medium term. In the longer term, stakeholders should establish holistic standards, guidelines, and frameworks that are globally accepted for better monitoring, measurement, and control of carbon footprints.

There are certifications—such as Leadership in Energy and Environmental Design—that could be included in standards or extended to achieve energy-efficient cloud datacenters. The research community needs to promote its work to make computation, networks, storage, and computer usage as energy efficient as possible. Policymakers should also advocate for strong policies that can force carbon footprint reduction in a cloud environment.

Acknowledgments

The authors thank Infosys for providing the opportunity and work environment to produce this article. The views expressed here are the authors' personal opinions; they might not represent the view of Infosys.

References

1. A Berl et al., "Energy-Efficient Cloud Computing." *The Computer Journal*, vol. 53, no. 7, 2009, pp. 1045–1051.
2. J. Baliga et al., "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport," *Proc. IEEE*, vol. 99, no. 1, 2010, pp. 149–167.
3. "Make IT Green—Cloud Computing and its Contribution to Climate Change," Greenpeace, 30 Mar. 2010; www.greenpeace.org/usa/en/media-center/reports/make-it-green-cloud-computing.
4. S. Murugesan, "Harnessing Green IT: Principles and Practices," *IT Professional*, vol. 10, no. 1, 2008, pp. 24–33.
5. F.-S. Chu, K.-C. Chen, and C.-M. Cheng, "Toward Green Cloud Computing," *Proc. 5th Int'l Conf. Ubiquitous Information Management and Communication* (ICUIMC 11), ACM, 2011, pp. 31:1–31:5.
6. R. Buyya, A. Beloglazov, and J.H. Abawajy, "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges," *Proc. 2010 Int'l Conf. Parallel and Distributed Processing Techniques and Applications* (PDPTA 10), ACM, 2010, pp. 6–20.
7. A. Younge et al., "Efficient Resource Management for Cloud Computing Environments," *Proc. Int'l Conf. Green Computing* (GreenComp 10), IEEE CS, 2010, pp. 357–364.
8. A. Beloglazov and R. Buyya, "Energy Efficient Allocation of Virtual Machines in Cloud Data Centers," *Proc. 2010 10th IEEE/ACM Int'l Conf. Cluster, Cloud and Grid Computing* (CC GRID 10), IEEE CS, 2010, pp. 577–578.
9. K.H. Kim, A. Beloglazov, and R. Buyya, "Power-Aware Provisioning of Cloud Resources for Real-Time Services," *Proc. 7th Int'l Workshop on Middleware for Grids, Clouds and e-Science* (MGC 09), ACM, 2009, pp. 1:1–1:6.
10. H. Huang, "A Sustainable Systems Development Lifecycle," *Proc. 2008 Pacific Asia Conf. Information Systems* (PACIS 08), AISel, 2008, paper 81; http://aisel.aisnet.org/pacis2008/81.
11. J. Rott, "Developing Green Software," white paper, Intel, June 2011; http://software.intel.com/en-us/articles/developing-green-software.
12. K.S. Vallerio, L. Zhong, and N.K. Jha, "Energy-Efficient Graphical User Interface Design," *IEEE Trans. Mobile Computing*, vol. 5, no. 7, 2006, pp. 846–859.
13. N.R. Potlapally et al., "Analyzing the Energy Consumption of Security Protocols," *Proc. of the 2003 Int'l Symp. Low Power Electronics and Design* (ISLPED 03), ACM, 2003, pp. 30–35.

Nitin Singh Chauhan *is a senior research scientist at Infosys, India. His research interests include cloud computing, information security, and sustainable IT. Chauhan received his masters in computer application from Jai Narain Vyas University, Jodhpur, India, and he has Certified Information Systems Security Professional (CISSP) and Certified Information Systems Auditor (CISA) certifications. Contact him at nitin_chauhan01@infosys.com.*

Ashutosh Saxena *is a principal research scientist at Infosys, India. His main research interest is information security. Saxena received his PhD in computer science from Devi Ahilya University, Indore, India. He's a senior member of IEEE. Contact him at ashutosh_saxena01@infosys.com.*