# Deploying and Provisioning Green Software

**Ayse Basar Bener and Andriy Miranskyy**, Ryerson University

**Steve Raspudic**, IBM Canada

*// To learn more about how software developers can integrate green software practices, the guest editors spoke with Steve Raspudic, manager and deployment and provisioning architect at IBM Toronto's Software Lab. //*

**Are mainstream developers concerned with the greenness of their products?**

In general, people are aware of energy-efficiency concerns—for example, they try to remove unused hardware. And a lot of energy-efficiency concerns are driven by density concerns: people are trying to maximize hardware utilization using virtualization technology. They aggregate multiple workloads on a single computer (at least on the server side) so that workloads and software stacks can run on smaller amounts of hardware. What people are doing now is making sure that hardware is fully utilized, which is now achieved primarily via virtualization. In terms of actual design of software, we aren't there yet, and I think there isn't as much focus on that. At this stage, the key to energy efficiency is virtualization: either in a public or private cloud, at least for server-side workloads.

**How can we promote green software values?**

People are subconsciously thinking, "If I build more efficient software, I'll consume less energy." We need a global standard that's bigger than any practitioner who promotes it. Otherwise, it's going to be really tough to raise awareness. For businesses to promote these values, there should be some bottom-line advantage. Historically, we've seen that this can be achieved by using benchmarks. Classic benchmarks are typically easy to understand and measure directly—for example, you can directly measure the number of transactions per unit of time in relational databases. However, electricity measurement is more complex—it's hard to measure electricity consumption associated with a specific workload on a production system.

Power-efficient benchmarks do exist. For example, TPC-Energy is designed to measure the energy efficiency of database engines. However, they haven't been widely adopted by the industry. We need industry bodies to drive these benchmarks. Potentially, a regulatory body—pushing for adoption of energy benchmarks—backed up by major players in the market could help achieve this goal.

Recently, studies have been published on the average electricity consumption of a single Google search query. These types of studies help raise the general public's awareness of software greenness. This can help get public support for building such a regulatory body.

**Do you think that it's important to consider greenness of software during its design?**

Yes, energy efficiency should be more important. But I would say that it should come naturally from striving for the general efficiency that people are striving for in software business. It comes down to general algorithm efficiency, where you want to make thing run fast with less hardware. This is now amplified and driven by pricing schemas of cloud resources

 0740-7459/14/$31.00 © 2014 IEEE

and cost savings: when you rent hardware on the cloud, you pay less if you consume fewer resources. To minimize the cost of virtual hardware, you need to write efficient software that can do more with limited resources. This will translate into power efficiency because you increase the amount of work done per CPU cycle.

**But lower cost doesn't necessarily mean greener... .**

There are exceptions, but you can argue that reduction of cost in most cases will translate into energy savings. That's something interesting to research further. One of the big cost factors for datacenters is power, in addition to hardware, staffing cost, and so on. If you see that your hardware rental bill is going down, very likely it translates into energy reduction. We'll get energy saving from economic considerations: people will try to optimize their software to lower their monthly bill. Note that the amount of effort invested into the optimization would depend on what percentage of the monthly bill is driven by electricity consumption.

**How would energy efficiency translate into better design?**

It comes down to the software's overall runtime efficiency, which will translate into lower energy costs. Indirectly, the greenness will come.

**Is there a pressure from customers to build more energy-efficient software?**

Not directly. The main pressures are to reduce cost and increase capabilities. But we'll get there indirectly. In our products [software-hardware appliances], we're promoting the concept of density, which means that you can run a lot of different workloads on the same rack of hardware. This translates indirectly into power efficiency.

**How do you see the evolution of green software?**

For example, in the Web server domain, a lot of Web applications are written in interpreted languages because the code is easy to build and maintain, reducing time to market. However, these languages aren't very efficient, requiring a lot of hardware resources. As Web applications evolve, the key parts of the application should be rewritten in compiled languages to reduce runtime cost. What we're seeing now is a return of more efficient languages for server-side applications: compiled languages are making a big comeback. Again, energy efficiency is not what's driving them, but indirectly it will lead them to energy usage reduction.

On the other hand, a lot of system-level software is already as efficient as it can get, maximizing the amount of work, while minimizing CPU, memory, and disk consumption.

**In a legacy system, if a new requirement has to be implemented, it's difficult to change the underlying language. Do you address hardware constraints at IBM?**


## ABOUT THE AUTHORS

**AYSE BASAR** Bener is a professor in and the director of the Data Science Lab at Ryerson University. Contact her at ayse.bener@ryerson.ca.

**ANDRIY MIRANSKYY** is an assistant professor in the Department of Computer Science at Ryerson University. Contact him at avm@ryerson.ca.

**STEVE RASPUDIC** leads the provisioning team for the PureData appliance at IBM Canada. Contact him at steven.raspudic@gmail.com.

We're in a sense of, "Can we implement a new feature without increasing existing hardware capacity?"

There's also a trade-off between time to market and runtime efficiency. Software often ships really quickly just "to get it out there." Then, as time progresses, the runtime can be optimized in subsequent releases. The key goals are to ship products quickly and efficiently enough to meet the service-level agreements to which developers are committed. But there's no general desire to be as efficient as possible, at least for the initial product release. Generally, the first release will do the least possible for software to generate revenue. Then, over time, the software gets optimized.

If a product is unsuccessful, there'll be minimal investments in its subsequent releases. However, due to lack of success, a wide customer base won't use the product. Therefore, the product's overall energy consumption will be small.

**The green movement uses $CO_2$ emissions metrics extensively. Will software developers adopt these types of metrics or won't they resonate with software practitioners?**

It's difficult to compute $CO_2$ emissions for a general software product. These metrics are taken into consideration during datacenter design and construction. However, if we can do more on that, it would be useful—for example, determining a Google search's $CO_2$ emission.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.