

Low-Energy Software?

By Peter Mezzina

March 20, 2009

Consider a basic relationship: software performance engineering makes it possible to run the same functionality on the same hardware at faster response times. Now imagine what might result from performance engineering taken beyond current maximum allowable response times specified by service agreements: software designed to fulfill the same functionality with the same response times on less hardware. We will see how this innovation would translate to substantial savings in energy and operating costs.

Current Trends

We are already moving in the direction of greater data center energy efficiency. Consider the following trends.

Power management and other efficiencies allow hardware manufacturers to comply with the US EPA EnergyStar standard. Already broadly supported for the desktop this standard will increasingly apply to the data center where hardware will inevitably use less energy.

Virtualization makes allocation of hardware resources to software more efficient and in turn reduces the amount of hardware needed for a system. With virtualization the key concept is finding resources that are underutilized and by virtualizing software borrow those underused resource for processing that would otherwise overwork other resources. Companies such as VMWare have had great success with this approach, mostly for operational cost-cutting but with parallel reductions in power usage.

What's Next

The missing piece resides within the software itself, in particular the processes used to design it. To bring about energy efficient software we would not need much change to re-tool the way we develop today. Consider a typical development team anywhere in the world. The group manages three quantities: team size budget limits, project time limits and extent of functionality. When budget cannot be added further and functionality cannot be removed, time becomes the only remaining variable. As project time passes and a deadline looms developers are forced to make compromises, particularly non-functional requirements like the processing efficiency of software. Performance requirements such as response times are met to their minimal level and then further performance-tuning is dropped. When that continued effort is dropped, energy efficiency is dropped with it. In my experience the lost efficiency can be considerable.

Today, there is no widely-accepted method for incorporating energy efficiency into a software design exercise. A simple means for doing this would have huge impact. Consider:

- In 2005 the US consumed a total of 40 Billion Kilowatt-Hours of electric within data

centers. A minimum of 20% of this usage resulted from inefficient software architecture choices made in the design phases of the software development process.¹

- No established methodology helps engineers make energy efficient software design choices.
- A 20% reduction of server usage would translate to \$1.2 billion saved annually on electric alone, and data center electric consumption continues to double every five years. By reducing servers through these efficiencies, savings on non-server hardware, facilities and administration would minimally triple that savings.

Were such a method in use it would provide numerous benefits beyond the obvious:

- For corporations it would reduce:
 - Electric costs
 - Hardware purchases, including servers, storage, network devices
 - Administrative costs
- For highly visible public websites (Google, Yahoo) energy efficiency would attract user loyalty, much like the EnergyStar program does for hardware vendors.
- For government, national energy consumption could be reduced through a optional compliance standard like the EnergyStar program.

My own effort to write a disciplined energy-efficiency design method (Small Footprint Design Method) focuses on:

- having a disciplined set of design steps to be used in the design phase to enable software performance engineering in general but potentially for energy efficiency gains
- minimizing impact on the cost of design and
- developing the added benefit of an ROI that can be estimated fairly well.

A disciplined and measurable design method for software performance engineering would build confidence among decision-makers considering energy efficiency. Beyond cost containment, energy-efficiency will increasingly create marketing goodwill for an organization.

Peter Mezzina is an ITAC Certified IT Architect and the founder of Process Intelligence (<http://www.procint.com>), an 11-year-old firm providing IT architecture services. He can be e-mailed at pmezzina@procint.com.

¹ JG Koomey, May 2008, *Worldwide Electricity Used In Data Centers*, Lawrence Berkeley National Laboratory and Stanford University