# Energy-Aware Programming in Pervasive Computing

Yu David Liu
Assistant Professor
Department of Computer Science
State University of New York at Binghamton
davidL@cs.binghamton.edu

What will mainstream computing look like in 2020? It is perhaps no longer daring to imagine a picture with complex interactions among millions of mobile devices, thousands of sensor nodes, and hundreds of cloud services. The key players in this picture – mobile devices, sensors, and clouds – are invariantly linked to energy consumption. Mobile devices – laptops, smartphones, and tablets – are mobile only when they function over battery, a finite source of energy. Sensor networks are often deployed in remote/hostile/inconvenient areas where recharging battery is impossible or expensive. For cloud service providers, power consumption not only translates to electricity bills, but also affects the reliability of their services due to accompanying heat dissipation.

Existing research on achieving energy efficiency largely focuses on innovations of VLSIs, architectures, operating systems, and compilers. This white paper calls for more investigations from the perspective of *programming language design*, *i.e.* how new programming models can help construct energy-aware software, and foster next-generation green-conscious programmers. Achieving energy efficiency from the level of programming models (as apposed to systems or hardware) has some unique benefits:

- *Effectiveness with Application-Level Inputs.* Programming languages are the interface between programmers and computing platforms – the epicenter of "human-computer interaction" in the development lifecycle. A more energy-friendly language design increases programmer awareness of energy consumption, and promotes application-level energy-efficient solutions directly from programmers. As an example, a programmer intuitively knows that running a GPS localization program with a precision of 0.9995 is likely to consume more energy than the same program with precision 0.5. An ideal language design should provide easy-to-use language abstractions to encourage the use of precision 0.5 when the GPS device's energy level is low.

- *Portability.* An essential trait of pervasive computing is computations often happen across heterogeneous platforms. Programming language solutions excel at providing platform-independent abstractions, which can in turn be mapped to different platforms by compilers, middleware, and virtual machines.

In the workshop, the author plans to discuss an ongoing project, a lightweight program annotation system for energy-aware programming. In this system, energy states can be abstracted as programmer-defined modifiers, and individual code fragments can be labeled with expected energy states appropriate for their executions. The system can be used for several purposes:

- the modifiers can be used to make system/hardware-level decisions, such as energy-efficient/aware scheduling and dynamic voltage and frequency scaling.
- a program analysis can be constructed to rigorously reason energy consumption compositionally, and check the consistency of programmer intentions.

The author is a researcher in programming languages and compilers. This white paper is aligned with his current interest in energy-aware programming [3]. In addition, he is interested in designing programming languages for sensor networks [5] and clouds [4], the domains important for building next-generation pervasive computing platforms. His recent work on parallel languages coined the phrase *pervasive atomicity* [1], a correctness property for multi-threaded software with the strength for scalability. His doctoral dissertation [2] centered around providing language support for modeling complex interactions in modern ultra-large-scale software, a theme relevant to the workshop.

# References


[1] Kulkarni, A., Liu, Y. D., and Smith, S. F. Task types for pervasive atomicity. In *OOPSLA '10: Proceedings of the 25th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (Reno, NV, USA, 2010).

[2] Liu, Y. D. *Interaction-Oriented Programming.* PhD thesis, Johns Hopkins University, Baltimore, MD, USA, 2007.

[3] Liu, Y. D. Programming models for green software. NSF CAREER Award, 2011.

[4] Liu, Y. D., and Gopalan, K. Interaction-based programming towards translucent clouds. In *APLWACA '10: Proceedings of the 2010 Workshop on Analysis and Programming Languages for Web Applications and Cloud Applications* (Toronto, Canada, 2010), pp. 15–19.

[5] Liu, Y. D., Skalka, C., and Smith, S. Type-specialized staged programming with process separation. *Under Revision for Journal of Higher-Order and Symbolic Computation* (2010).