Challenges in Continuous Sensing from Heterogeneous Metadata Sources

(Position Paper)

Alex Doboli, Tom Robertazzi and Sangjin Hong
Department of Electrical and Computer Engineering
State University of New York at Stony Brook
Email: adoboli@ece.sunysb.edu

Abstract

Next-generation supervisory and process control (SPC) systems must offer efficient and secure utilization of critical infrastructure, provide less costly protection of the environment, and assure a safer environment for living. Advanced SPC systems are based on continuous sensing, processing, storing, and communication of high-level data (metadata). Traditional embedded platforms offer modest execution support for metadata processing, as they are too slow, unreliable, and energy hungry. This is a major obstacle in designing and deploying more intelligent and autonomous SPC systems, and the further development of intelligent control. We argue that embedded systems must be changed from reactive but passive operation to an active, anticipative operation.

1 Introduction

Next-generation supervisory and process control (SPC) systems must offer a more efficient utilization of the nation's critical infrastructure (e.g., roads, buildings, power grid), provide less costly protection of the natural world, and in general, assure a safer and more pleasant environment for living. Advanced SPC systems are based on the (i) acquisition of high-level data (metadata), such as video images, sound, olfactory, and hyperspectral images, (ii) processing, storing, and communication of metadata, and (iii) intelligent control using higher-order semantics extracted from metadata [1, 2, 3, 4, 5]. Data is continuously collected from the environment to discover and plan new activities of interest, or to observe general trends (patterns) about the monitored situation. Then, the learned knowledge is used to provide self-optimizing responses, including faster and more intelligent anticipation and reaction to critical events, superior utilization of the monitored infrastructures, and better survivability in harsh situations. The broader vision is to transform embedded systems' behavior from a reactive but passive operation (only in response to external events) to an active, anticipative operation.

Traditional embedded platforms (including hardware architectures and system software) offer modest execution support for active, anticipative algorithms, as they are too slow, unreliable, and energy hungry. Anticipative algorithms involve computationally intensive techniques for low-level metadata processing (e.g., filtering, feature extraction) and high-level analysis (like identification, Gaussian mixture models, particle filters, motion estimators, learning, etc.) [1, 2]. The lack of a well-suited embedded platform is a major obstacle in designing more intelligent and autonomous SPC systems, and in the further developing of the theory on intelligent control. A naive approach would aggressively utilize high performance processors and networks, and optimize the system software for the worst case situation. However, this results in very expensive systems, likely to be unattractive in real life. Moreover, high performance processing is very energy hungry, hence unsuitable for mobile, autonomous systems. The battery life of such systems is very short, and present energy harvesting methods are not capable to deliver the energy needed for metadata processing. Moreover, the projection is that the complexity of future metadata processing and analysis will continue to increase, thus, making operation increasingly inefficient with embedded platforms oriented only on reactive, statically-optimized processing.

We argue that implementing active, anticipative processing requires a new paradigm (including the related formalism and algorithms) for developing system software that manages the underlying hardware architecture and energy resources. In traditional embedded systems, the system software optimizes the usage of a fixed set of resources (processing hardware, communication bandwidth, and supply energy) functioning in static conditions [6]. Architectures have a fixed structure and organization, and their performance can be predicted using static models. In contrast, the functionality and performance constraints of SPC systems are far less predictable

as the characteristics of the monitored environment are continuously changing. Also, the amount of resources available to a given task depends on the computational demands of other tasks, as all tasks share the same hardware and battery energy. The dynamics of processing and performance needs does not follow any particular mathematical rule, being rather a mixture of different types of behavior, e.g., quasi-static, probabilistic, and performance constrained operation [2, 7, 8]. Therefore, the system software for active, anticipative processing in SPC systems must continuously "comprehend" the new processing demands, and accordingly customize the hardware architectures and optimize on-line the dispatching of resources to tasks.

There are several interesting attempts going on for improving hardware usage through on-line learning and adaptation [9, 10, 11, 12]. The work focuses mostly on large computing systems, like servers. Experimental evidence shows that system adaptation leads to better performance, such as higher throughput due to better load balancing and resource availability [13, 14, 12, 15], or better distribution of cache memory to resources [16, 11]. On-line architecture adaptation is achieved through methods inspired from adaptive control theory [17, 10, 18, 19, 20, 21, 22], evolving recurrent neural networks for adaptation through reinforcement learning [11], machine learning [16], and rule based adaptation [12]. Learning uses low level statistics collected about the system workload [16, 2, 7, 12] (i.e. information about memory usage and swapping, context switches, interrupts, and CPU utilization), but also more complex mechanisms, like Tree Augmented Bayesian Networks [9], for correlating system metrics with the performance of the building blocks.

2 Main Characteristics of Embedded Metadata Processing

This section summarizes the main features of metadata processing in real environments. It uses camera-based target recognition and tracking as an illustrating example.

A. System functionality. Images are periodically sampled using an embedded camera, after which they are locally processed to isolate objects of interest (like human faces or moving vehicles). Then, compressed information about objects is sent through the wireless network to other nodes or to the server. Nodes are powered by batteries. The transmission of raw images to the server is not an option, as wireless transmission consumes about $100\times$ to $1,000\times$ more energy than local processing. In addition, modern VLSI fabrication processes offer high integration densities that basically guarantee that minuscule silicon areas can implement powerful computing and communication resources. Compared to nodes with simple sensing and no/minimal signal processing, camera-based sensor nodes are more powerful in tracking objects of different kinds. After all, a simple proximity sensor cannot differentiate between a truck and a compact car.

Identifying objects of interest in an arbitrary image is a fundamental step in metadata processing. The computational complexity of these algorithms is huge due to the large amounts of image data to be processed. Execution time of object detection is about 1 million instruction cycles, which requires a programmable processor operating at around 8GHz, if a moving object traveling at a reasonable speed were to be tracked by sampling of each traveled meter. However, this solution is not feasible as the lifetime of batteries would be short (due to the large speed), and the cost of the system high. In addition, the system would be ill-designed for slower or faster moving objects, leading to either waste of hardware and energy resources, or to poor tracking.

B. Dynamics of performance requirements. SPC systems operate in environments with changing properties, like the number and kind of objects of interest, available energy resources, temperature, communication bandwidth, and so on. If there are no moving objects, then the throughput of each node has to be kept low, so that its energy consumption is reduced. As moving objects approach, the throughput of nodes must continuously increase for having more image samplings per second. The highest throughput requirement occurs when moving objects pass by the vicinity of a node. Then, the throughput requirement might be much higher than during idle intervals. Performance requirements change dynamically being considerably different from typical embedded applications that operate in static or quasi-static conditions. Optimizing the system for the worst case wastes important hardware and energy resources.

SPC nodes are organized into ad-hoc networks of varying topologyc. Topologies continuously change not only because of adding or removing nodes, but also because nodes dynamically cluster together to split the com-

putational burden. The performance constraints of individual nodes, e.g., throughput, are affected by the positioning of nodes in space, as well as the continuous evolution in time of the network topology. For example, close sensor nodes cluster together to share the load of image processing at high sampling rates. Because of physical proximity, the energy used for communications inside the cluster remains low. Reacting to different performance gradients implies that SPC node architectures must gracefully adapt without wasting hardware and energy resources.

3 Research Needs

We argue that implementing active, anticipative processing requires a new paradigm (including the related formalism and algorithms) for developing system software that manages the underlying hardware architecture and energy resources. In contrast to traditional embedded systems, the functionality and performance constraints of SPC systems are far less predictable as the characteristics of the monitored environment are continuously changing. Also, the amount of resources available to a given task depends on the computational demands of other tasks, as all tasks share the same hardware and battery energy. The dynamics of processing and performance needs does not follow any particular mathematical rule, being rather a mixture of different types of behavior, e.g., quasi-static, probabilistic, and performance constrained operation. Therefore, the system software for active, anticipative processing in SPC systems must continuously "comprehend" the new processing demands, and accordingly customize the hardware architectures and optimize on-line the dispatching of resources to tasks.

Active, anticipative processing requires a new system software paradigm that offers low overhead performance requirement identification and comprehensive adapting of architectures to continuously changing environments.

1. *Low adaptation overhead*: Existing algorithms require very large overhead in terms of computing time, energy, and hardware resources, which - often, nullifies the benefits of adaptation. For example, in quickly changing environments, the learning time is longer than the time window it can predict, thus, learned attributes continuously lag behind real values. Also, adaptive control cannot "re-use" the sequence of adaptation steps for a situation encountered before, thus wasting overhead by repeating the same sequence.

2. *Comprehensive adaptation*: Embedded architectures have a variety of architectural "knobs" through which they are customized to specific needs. Knobs include dynamic digital reconfiguration, reconfigurable analog blocks, adjustable bus systems, configurable I/O systems, dynamic voltage and frequency scaling, and many more. Present adaptation and control policies are limited in comprehensive tackling the dispatching of various resources to sampling, processing, and networking tasks.

3. *Formalisms for dynamic systems*: There are few design-oriented formalisms capable of expressing not only the functionality of a system but also the dynamics of its performance. Existing formalisms are mostly implicit descriptions, and have to be simulated for understanding the system dynamics. This is inconvenient for anticipative, on-line system optimization, as understanding the quality of a certain architectural modification demands spending important resources and energy.

References

[1] U. Anliker, J. Beutel, M. Dyer, R. Enzler, P. Lukowicz, L. Thiele, and G. Troster. A systematic approach to the design of distributed wearable systems. *IEEE Transactions on Computers*, (53)8:1017–1033, 2004.

[2] T. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranov, and V. Eruhimov. Computer vision workload analysis: Case studt of video surveillance systems. *Intel Technology Journal*, 9(2):109–118, 2005.

[3] S. Kyo, T. Koga, S. Okazaki, and I. Kuroda. A 51.2-gops scalable video recognition processor for intelligent cruise control based on a linear array of 128 four-way vliw processing elements. *IEEE Journal of Solid-State Circuits*, 38(11):1992–2000, 2003.

[4] A. Gentile and S. Wills. Portable video computing. *IEEE Transactions on Computers*, (53)8:960–987, 2004.

[5] A. Lodi, M. Toma, F. Campi, A. Capelli, R. Canegallo, and R. Guerrieri. A vliw processor with reconfigurable instruction set for embedded applications. *IEEE Journal of Solid-State Circuits*, 38(11):1876–1886, 2003.

[6] R. Ernst. Codesign of embedded systems: Status and trends. *IEEE Design & Test*, pages 45–54, 1998.

[7] F. Wang, Q. Xin, B. Hong, S. Brandt, E. Miller, and D. Long. File system workload analysis for large scale scientific computing applications. In *IEEE Conference on Mass Storage Systems and Technologies*, 2004.

[8] S. Brandt. Performance analysis of dynamic soft real-time systems. In *IEEE International Performance, Computing, and Communication Conference*, 2001.

[9] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. Chase. Correlating instrumentation to system states: A building block for automated diagnosis and control. In *6th Symposium on Operating Systems Design and Implementation (OSDI)*, 2004.

[10] Y. Diao, J. Hellerstein, S. Parekh, and J. Bigus. Managing web server performance with autonomous agents. *IBM System Journal*, 42(1), 2003.

[11] F. Gomez, D. Burger, and R. Miikkulainen. A neuroevolution method for dynamic resource allocation on a chip multiprocessor. In *Proc. of International Joint Conference on Neural Networks*, pages 2355–2361, 2001.

[12] J. Wildstrom, P. Stone, E. Witchel, R. Mooney, and M. Dahlin. Towards self-configuring hardware for distributed computer systems. In *Proc. of the International Conference on Autonomic Computing*, pages 241–249, 2005.

[13] S. Brandt, G. Nutt, T. Berk, and J. Mankovich. A dynamic quality of service middleware agent for mediating application resource usage. In *IEEE Real-Time Systems Symposium*, pages 307–317, 1998.

[14] G. Nutt, S. Brandt, A. Griff, and S. Siewert. Dynamically negotiated resource management for data intensive application suites. *IEEE Transactions on Knowledge and Data Engineering*, 12(1):78–95, 2000.

[15] Q. Wu, V. Reddi, Y. Wu, J. Lee, D. Connors, D. Brooks, M. Martonosi, and D. Clark. A dynamic compilation framework for controlling microprocessor energy and performance. In *International Symposium on Microarchitectures*, pages 271–282, 2005.

[16] I. Ari, A. Amer, R. Gramacy, E. Miller, S. Brandt, and D. Long. Acme: Adaptive caching using multiple experts. In *Distributed Data and Structures, 4*, pages 143–158. Carleton Scientific, 2002.

[17] Y. Bai and I. Bahar. Reducing issue queue power for multimedia applications using a feedback control algorithm. In *International Conference on Computer Design*, 2004.

[18] Y. Zhu and F. Mueller. Feedback edf scheduling exploiting dynamic voltage scaling. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 84–93, 2004.

[19] C. Lu, T. Abdelzaher, J. Stankovic, and S. Son. A feedback control approach for guaranteeing relative delays in web servers. In *IEEE Real-Time Technology and Applications Symposium*, 2001.

[20] Z. Lu, J. Lach, M. Stan, and K. Skadron. Reducing multimedia decoder power using feedback control. In *International Conference on Computer Design*, pages 489–496, 2003.

[21] R. Zhang, C. Lu, T. Abdelzaher, and J. Stankovic. Controlware: A middleware architecture for feedback control of software performance. In *International Conference on Distributed Computing Systems*, 2002.

[22] A. Varma, B. Ganesh, M. Sen, S. Choudhury, Srinivasan L., and B. Jacob. A control-theoretic approach to dynamic voltage scheduling. In *International Conference on Compilers, Architectures, and Synthesis for Embedded Systems*, 2003.