Computational and Compressed Sensing Optimizations for Information Processing in Sensor Network

VIJAY KUMAR
University of Missouri-Kansas City, Kansas City, Missouri, USA

1. INTRODUCTION

This report narrates the development and deployment of a Distributed Sensing Algorithm that had a profound impact on the performance and capabilities of a number of computing systems. It solved a number of complex issues in the deployment of large scale sensor networks. We begin with a brief history of the development of this algorithm.

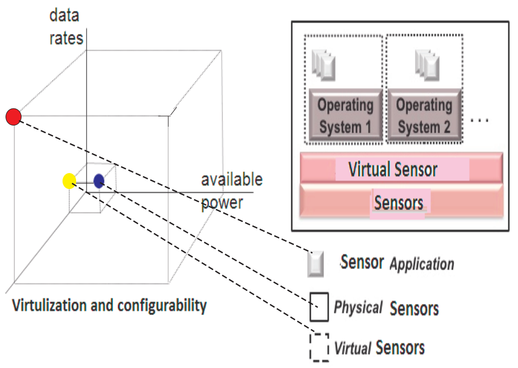

Figure 1 : Real-Time extension of Brook-Iyengar algorithm showing Virtual sensor implementation.

In 1996, Iyengars group, in collaboration with Brooks and with funding from Oak Ridge National Laboratory, invented a method of fault tolerance modeling that offered a computationally inspired real-time task management solution. The algorithm referred to as BrooksIyengar algorithm or BrooksIyengar hybrid algorithm is a distributed algorithm, that improves both the precision and accuracy of the measurements taken by a distributed sensor network, even in the presence of faulty sensors. The algorithm does this by exchanging the measured value and accuracy value at every node with every other node. In addition, it computes the accuracy range and a measured value for the whole network from all of the values collected. The algorithm demonstrated that even if some of the data from some of the sensors is faulty, the sensor network does not malfunction.

The group, in their investigation observed that mapping a group of sensor nodes to estimate it value accurately needs all the sensors to exchange the values to each other making it computationally expensive. In their framework the sensors array that are highly distributed containing individual sensors measure a common phenomenon. The real-time sensor stream is sent to a virtual sensor which aggregates the setpoint only from good sensors. If the sensors are able to communicate within themselves then some of the redundant information can be eliminated, but such corporation would be highly energy inefficient. Design of such distributed systems would not know how many faulty sensors are present, so the use of Byzantine algorithm allows to solve this gap.

In 2000, the DARPA program manager used two major demonstrations to showcase SensITs advances and document the ability of sensor networks to provide new capabilities. One demonstration took place at the Twentynine Palms, California Marine Training grounds in August 2000, the other took place at BBN offices in Cambridge, Massachusetts in October 2011. Dr. Gail Mitchell of BBN coordinated this work for BBN, DARPAs SensIT integration contractor.

The Figure 1 shows the relationship to data rate, energy efficient and virtualization in a practical distributed sensor networks. Iyengar algorithm for noise tolerant distributed control to the fault-event disambiguation problem in sensor-networks is shown with a small circle connected to rectangle labeled "Virtual Sensors".

2. REAL-TIME EXTENSIONS

This work emerged in new versions of real time extensions [Rogina et al. 1987] to Linux Operating Systems by Pablo J. Rogina and Gabriell Wainer in 2001. Many of these algorithms were used and installed in the RT Linux Operating System. They are now working on formal model verification by incorporating the algorithms into a new embedded kernel for robotic applications. The profound contribution of the Brooks-Iyengar Distributed Computational Sensing work has enhanced new real-time features by adding fault tolerant capabilities.

2.1 MINIX Testbed

The MINIX 1.5 operating system [Brooks et al. 1998] was taken as a base, and it was extended with several real-time services [Chakrabarty et al. 2002]. The most important include task management capabilities (both for periodic an aperiodic tasks) and real-time scheduling algorithms (Rate Monotonic and Earliest Deadline First). These strategies were later combined with other traditional strategies, such as Least Laxity First, Least Slack First and Deadline Monotonic. At present new flexible schedulers are being included.

To allow these changes several data structures in the operating system were modified (to consider tasks period, execution time and criticality). A new multi-queue scheme was defined, so as to accommodate real-time tasks along with interactive and CPU-bound tasks. The original task scheduler of MINIX used three queues, in order to handle task, server and user processes in that order of priority. Each queue was scheduled using the Round Robin algorithm.

A new set of signals was added to indicate special situations, such as missed deadlines, overload or uncertainty of the schedulability of the task set. All these services were made available to the programmer as a complete set of new system calls. A long list of tests demonstrated the feasibility of MINIX as a workbench for real-time development.

Several works were done using the tool, spanning from the testing of new scheduling algorithms to kernel modifications. Recently, the need to integrate the previous work in a new version for the operating system became apparant. This was motivated in part for the release of new MINIX versions in the meantime, and because several additional features were identified that would be useful to be added to original environment.. Those extensions [Krishnamachari et al. 2004] were done using MINIX 2.0; include the previous services and add new ones such as analog-digital conversion, queue model modification and new real-time metrics. These services are described in detail in the following paragraphs.

The need to acquire analogic data from the environment motivated this new feature. As stated before, many real-time systems are used to control a real process, such as a production line or a chemical reaction. A device driver was written following the same framework used under Linux [Iyengar et al. 2010], with slight changes. The device driver adds a new kernel task that provide the programmer with three basic operations (open, read, close) to access an A/D converter as a character device (for instance, /dev/js0 and /dev/js1, for joystick A and joystick B, respectively).

A second set of changes was related with the task scheduler management. The ready process queuing and handling is arranged in four levels. The basic idea considered in joining the queues was related with the goal that a real-time task should not be interfered by low level interrupts (and its associated servers), working with the hypothesis that server and user queues can be joined, allowing File System (FS) and Memory Manager (MM) processes to be moved from server to user process category. An in-depth analysis was made to check the possibility of deadlock between FS and MM, first revisiting the semantics of them and then trying to measure the impact of the new scheduler (with the joined queues), showing that deadlocks cannot occur with the changed scheduler.

Once the OS was extended with real-time services, the need arose to have several measuring tools. It is needed to test the evolution of the executing tasks according with the different scheduling strategies. The impact of the different workloads should be also considered. To do so, the kernel is in charge to keep a new data structure accessible to the user via a system call. Statistics also can be monitored online by means of a function key displaying all that information.

MINIX proved to be a feasible testbed for OS development and real-time extensions that could be easily added to it. This "new" operating system (a MINIX 2.0 base with real-time extensions) has a rich set of features, which makes it a good choice to conduct real-time experiences. The added real-time services covered several areas:

Task creation: tasks can be created either periodic or aperiodic, stating their period, worst execution time and priority

Clock resolution management: the resolution (grain) of the internal clock can be changed to get better accuracy while scheduling tasks.

Scheduling algorithms: both RMS and EDF algorithms are supported, and can be selected on the fly.

Statistics: several variables about the whole operation are accessible to the user to provide data for benchmarking and testing new developments.

Supervisory Control and Data Acquisition: as a user application, it makes full use of real-time services.

2.2 Sense-IT

Sense-IT demonstrations used the Brooks-Iyengar fusion approach to combine sensor readings in real-time. Acoustic, seismic, and motion detection readings from multiple sensors were combined and fed into a distributed tracking system. The first deployment was effective, but noisy. The second demonstration built on the success of the first testing California. An improved outfielder algorithm was used to determine which node was best situated to continue existing tracks. This work was an essential precursor to the Emergent Sensor Plexus MURI from Penn State Applied Research Laboratory (PSU/ARL) with Dr. Shashi Phoha as PI. In that MURI, researchers from PSU/ARL, Duke, U. Wisconsin, UCLA, Cornell, and LSU extended SensIT's advances to create practical and survivable sensor network applications.

2.3 The Thales Group

The Thales Group, a UK Defense Manufacturer, used this work as part of its Global Operational Analysis Laboratory.

3. VIRTUALIZATION

The team found that the hardware support fails to provide an unambiguous performance advantage for two primary reasons: first, it offers no support for hardware fault detection; second, it fails to co-exist with existing software techniques for distributed sensor networks. They look ahead to emerging techniques as shown in Figure 1 for addressing this sensor virtualization problem in the context of software-assisted distributed fault-tolerance.

A real-time operating system can be modified to host the abstract fault-tolerant sensor layer with its own dynamic interval estimator, which is a mapping of the real-sensors.

An Abstract Sensor is a sensor that reads a physical parameter and gives out an abstract interval-estimate I, which is bounded and connected subset of the real line R.

A Correct Sensor is an abstract sensor where the interval estimate contains the actual value of the parameter being measured. If the interval estimate does not contain the actual value of the parameter being measured, it is called Faulty sensor.

4. COMPLEXITY OF ALGORITHM

As defined the abstract sensor finds an interval in a real-number line. The algorithm can sort the N sensor's data points along the x-axis, which can be performed in $O(\log(N))$ operations. Now moving from the highest value need to find the coarsest (L) to the finest (l) overlapping interval, which can be accomplished in $O(N)$ operations. Then The algorithm can estimate the interval $(l << L)$ for the virtual sensor in $O(N\log(N))$ operations.

REFERENCES

Brooks, R.R. and Iyengar, S. S., "Robust Distributed Computing and Sensing Algorithm" IEEE Computer. vol. 29 No. 6. pp. 53-60. June 1996.

Brooks, R.R. and Iyengar, S.S., "Multi-Sensor Fusion, Fundamentals and Applications with Software, 1998 Prentice Hall PTR.

Chakrabarty, K. Iyengar, S.S. H. Qi and E.C. Cho, Grid Coverage of Surveillance and Target Location in Distributed Sensor Networks, IEEE Transactions on Computers, Vol 51, No. 12, December 2002.

Krishnamachari, B. and Iyengar, S.S., "Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks", IEEE Tran Comp, 2004.

Iyengar et al, "Preventing Future Oil Spills with Software-Based Event Detection" IEEE Comp; 2010.

Rogina, Pablo J. and Wainer, Gabriel, "Extending MINIX with Real-Time Services and Fault Tolerance Capabilities" Infoteca, Departmentto de Computacion - FCEN, Argentina. 2000-2001.

Vijay Kumar is a Professor of Computer Science at the University of Missouri-Kansas City, Missouri, USA. He is the founding Editor-in-Chief of the International Journal of Next-Generation Computing (IJNGC). Prof. Kumar received his undergraduate and post graduate degrees in Physics, with specialization in Crystallography, from Ranchi University, India. He started his education in Computer Science in University of Cambridge, England, and completed his M.Sc. and Ph.D. degrees in Computer Science from the Universities of Manchester and Southampton in the U.K. He moved to the United States in 1983 and held faculty positions at the University of Cleveland, Ohio and University of Massachusetts, Boston before joining the University of Missouri-Kansas. He has published over 100 peer-reviewed papers in reputed international journals and conferences, besides 4 text-books and has been the Principal Investigator on 10 research projects from the Government and the Industry. His current research interests include mobile databases, web services, bio-informatics and sensor networks.