Object oriented machine learning with a multicore real-time java processor: short paper

Pedersen, Rasmus Ulslev; Schoeberl, Martin

Published in:

Proceedings of the 8th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2010)

Link to article, DOI:

[10.1145/1850771.1850782](https://doi.org/10.1145/1850771.1850782)

Publication date:

2010

Document Version

Early version, also known as pre-print

[Link back to DTU Orbit](#)

Citation (APA):

Pedersen, R. U., & Schoeberl, M. (2010). Object oriented machine learning with a multicore real-time java processor: short paper. In *Proceedings of the 8th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2010)* (pp. 76-78) <https://doi.org/10.1145/1850771.1850782>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Short Paper: Object Oriented Machine Learning with a Multicore Real-Time Java Processor

Rasmus Ulslev Pedersen
Embedded Software Laboratory
Department of Informatics
Copenhagen Business School
2000 Frederiksberg, Denmark
rup.inf@cbs.dk

Martin Schoeberl
Department of Informatics and
Mathematical Modeling
Technical University of Denmark
masca@imm.dtu.dk

ABSTRACT

The term intelligent systems is spreading beyond the data mining and machine learning communities. This presents new challenges that are fundamental to classical problems within object oriented programming and analysis. In this paper we investigate the use of a popular intelligent algorithm on a Java-based processor. The processor is a real-time enabled processor implemented on an FPGA, and we deploy a support vector machine on this processor. Furthermore, we show how this support vector machine can work on the Java-processor's multiple cores. This is a first step toward understanding how intelligent algorithms can be implemented on object-oriented Java systems with multiple cores in a hard real-time environment. Our experiments show significant speedup of the selected machine learning algorithm, and this can potentially be useful for other intelligent algorithms also.

1. INTRODUCTION

We explore machine learning on a multi-core version of the Java Optimized Processor (JOP) [6]. A field-programmable gate array (FPGA) can host up to 12 JOP cores. On each of those cores, we can install a machine learning algorithm. The intention is to speedup the machine learning and classification by parallelization on the multi-core system. The synchronization of the critical collection of data is a negligible part of the overall computational process.

Support vector machines (SVM) are part of a family of flexible machine learning algorithms for predicting structured objects and they are finding their way into mainstream computer science [2] and into embedded systems [3]. We use a standard dataset called Weather¹ to demonstrate the multicore-enabled SVM.

This paper describes a framework for using the SVM in a distributed environment with an emphasis on constrained

¹<http://www.cs.waikato.ac.nz/~ml/weka/index.html>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

JTRES'10 August 19–21, 2010 Prague, Czech Republic
Copyright 2010 ACM 978-1-4503-0122-0/10/08 ...\$10.00.

```
/**
 * Method getKernelOutput, which returns the kernel of two points.
 *
 * @param i1 - index of alpha_fp 1
 * @param i2 - index of alpha_fp 2
 * @return kernel output
 */
float getKernelOutputFloat(int i1, int i2) {

    kernelCalls++;

    return KFloat.kernel(i1, i2);
}
```

Figure 1: Normal Java code for the getKernelOutputFloat

computing [4].

Statistical learning theory has had a profound impact on learning theory over the last two decades, which is supported by the over 700 references at the SVM related site www.kernel-machines.org. Statistical learning theory has been developed and synthesized primarily by Vapnik [8]. He led the work on the support vector algorithm upon which this theory is based.

The paper is structured as follows. An overview of Support Vector Machines is presented in Section 2. We use the Java optimized processor (JOP) as the implementation platform, and this platform is introduced in Section 3. Experimental results are presented in Section 4, and the paper is concluded in Section 5.

2. SUPPORT VECTOR MACHINES

The SVM is an algorithm that is based mainly on work performed by Vladimir N. Vapnik and coworkers. It was presented in 1992 and has been the subject of much research since. We look at the algorithm from an application viewpoint and review its characteristics. A secondary purpose of this review is to introduce the definitions that play a central part in the following sections.

The SVM algorithm is a maximal margin algorithm. It seeks to place a hyperplane between classes of points such that the distance between the closest points are maximized. It is equivalent to maximum separation of the distance between the convex hulls enclosing the class member points. Vladimir Vapnik is respected as the researcher who primarily laid the groundwork for the support vector algorithm.

Figure 3: Optimization versus kernel counts

Cores	Classification
1	1,216,203
2	665,340
3	550,155

Table 1: Execution time in clock cycles for different number of cores on Weather data

The kernel code used to execute this example is introduced in Section 2. We can see how the kernel method in KFloat is converted to a control flow graph with basic blocks (code with no branches) and interconnecting vertices depicting the branches. Additional blocks are introduced for method invocations.

The SVM algorithm trains to optimize the objective function. The kernel² is a heavily accessed code section in any SVM. We have plotted the kernel counts versus the optimization function for the Weather data in Figure 3.

Using the JOP programs allow us to test the speed gains of using multiple processor cores. The experiments have been run in an FPGA platform with 1, 2, and 3 cores and a time-predictable memory arbitration scheme between the cores.

The speed gains on several cores, as shown in Table 1, are considerable. Doubling the number of cores gives almost a linear speedup. With three cores the bandwidth to main memory limits the additional speedup. This is an indication that more work on core local caches needs to be done on JOP. In general it is a sign that the communication overhead of running the multicore analysis is insignificant compared to the reduction in cycles of spreading the work across several processor cores. Figure 4 shows the speedup graphically.

5. CONCLUSION

In this paper we have taken the support vector machine algorithm, and demonstrated how it can be used in a Java multicore environment. The Java processor JOP is used as the execution platform, and the code is analyzed for the worst-case execution time. This is done for both a single CPU and multiple cores. The algorithm is parallelized on its most computational intensive points. We achieved linear scalability for two cores, and presented for the first time one of the most popular machine learning algorithms enabled

²a method called `getKernelOutputFloat`

Figure 4: Speedup on Weather data with different core configurations

for a real-time multicore execution environment. It is our conclusion that objected oriented intelligent algorithms are a subject of significant interest as we have demonstrated on popular family of algorithms on the JOP CMP processor.

6. REFERENCES

- [1] B. Huber. Worst-case execution time analysis for real-time Java. Master's thesis, Vienna University of Technology, Austria, 2009.
- [2] T. Joachims, T. Hofmann, Y. Yue, and C.-N. Yu. Predicting structured objects with support vector machines. *Commun. ACM*, 52(11):97–104, 2009.
- [3] R. Pedersen and M. Schoeberl. An embedded support vector machine. In *Proceedings of the Fourth Workshop on Intelligent Solutions in Embedded Systems (WISES 2006)*, pages 79–89, Vienna, Austria, June 2006.
- [4] R. U. Pedersen. *Using Support Vector Machines for Distributed Machine Learning*. PhD thesis, Dept. of Computer Science, University of Copenhagen, 2005.
- [5] C. Pitter and M. Schoeberl. A real-time Java chip-multiprocessor. *Trans. on Embedded Computing Sys.*, accepted for publication, 2010.
- [6] M. Schoeberl. A Java processor architecture for embedded real-time systems. *Journal of Systems Architecture*, 54/1–2:265–286, 2008.
- [7] M. Schoeberl and R. Pedersen. WCET analysis for a Java processor. In *Proceedings of the 4th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2006)*, pages 202–211, New York, NY, USA, 2006. ACM Press.
- [8] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, NY, 1995.