An extension for Transactional Memory in OpenMP

KEVIN OLIVEIRA[1]; ANDRE JARDIM[2]; GERSON CAVALHEIRO[3] ; ANDRE DU BOIS[4]

[1] Universidade Federal de Pelotas – kodoliveira@inf.ufpel.edu.br
[2]Universidade Federal de Pelotas– andre.jardim@inf.ufpel.edu.br
[3]Universidade Federal de Pelotas – gerson.cavalheiro@inf.ufpel.edu.br
[4]Universidade Federal de Pelotas – dubois@inf.ufpel.edu.br

1. INTRODUCTION

The Transactional Memory (TM) model has emerged as a promising alternative to synchronization mechanisms based on mutual exclusion. Transactional memory, which can be implemented in hardware or software, offers an alternative with a higher level of abstraction to classic synchronization mechanisms, providing a new synchronization control construct that avoids common blocking problems and significantly simplifies the programming effort to produce correct software (HARRIS; LARUS; RAJWAR, 2010). A strong argument for the employment of TM is based on its advantages in comparison with the use of mutex: scalability, composability, robustness and contention reduction as shown in HARRIS; MARLOW; PEYTON-JONES; HERLIHY (2005), SRIVATSA; KUMAR (2012) and WAMHOLFF; RIEGEL; FETZER; FELBER (2010).

Modern APIs (Application Programming Interface), such as OpenMP, TBB and even C++ when considered the concurrent programming resources introduced in recent standards, do not include natively a TM programming model. Some works in the literature, such as BAEK; MIHN; TRAUTMANN; KOZYRAKIS; OLUKOTUN (2007), MILOVANOVIC; FERRER; GAJINOV; UNSAL; CRISTAL; AYGUADÉ; VALERO (2007), WONG; AYGUADÉ; GOTTSCHLICH; LUCHANGCO; SUPINSKI; BIHARI (2014) and WONG; BIHARI; SUPINSKI; WU; MICHAEL; LIU; CHEN (2010), show efforts to achieve this inclusion through extensions. Commonly, these proposals present a transaction-handling interface that extends the one offered by OpenMP introducing a new specific directive to handle transactions. This approach can be seen as a great starting point for studying transactions within OpenMP.

The research carried out proposes to extend OpenMP incorporating a new memory layer, the TM, shared among tasks. This is done by incorporating a new feature in its programming interface allowing to bind variables to this memory layer. The obtained gain is to associate the semantics of operations on transactional memory with those offered by the other operations already available in OpenMP. The objective of the work is to extend the state of the art in interfaces for multithreaded concurrent programming by introducing resources for handling transactional memories in well established programming programming tools. The introduction of TM mechanisms in multithreaded programming tools is studied with regard to programming interface issues. It is understood in this work that its effective adoption can occur when its use is in line with the design decisions of the native programming interface. The proposed approach considers this issue, proposing programming features compatible with those already offered in OpenMP.

2. METHODOLOGY

The experimentation was carried out considering two sets of programs. The first is represented by the Cowichan, WILSON (1995), benchmark applications. Programs from this benchmark were implemented to validate the interface. The second set of programs consists of an application of the STAMP benchmark (bayes) and a program developed as a benchmark for OpenMP (kmeans). The experiments in steps 1 and 2 were executed using two NUMA platforms: Hydra (Opteron architecture with 64 cores, 4 nodes, 120 GB RAM) and Tekoha (Xeon architecture with 192 cores, 8 us, 120 GB RAM).

The performance index collected is the execution time, presented in seconds, obtained using an average of at least 30 executions in each case. Execution times for 2, 4, 8, 16, 32 and 64 threads were collected in the OpenMP execution team. In the presentation of these averages, it is highlighted when the samples of each case adhere to a normal distribution or not. For this, the Kolmogorov-Smirnov test was performed with 95% confidence. The comparisons between the cases use adequate statistical tests to obtain positioning of the performance between two cases. When samples adhere to a normal curve, Student's T was used, otherwise the Mann-Whitney U test was used, both also with 95% confidence.

3. RESULTS AND DISCUSSION

The proposed interface was validated regarding both its expression power and its performance. The expression power was validated by implementing six Cowichan benchmark programs in which the applicability of the transactional memory model was possible: hull, norm, outer, sor, thresh and vecdiff. The performance was evaluated by implementing kmeans and bayes applicaitions.

Three implementations were evaluated for the selected Cowichan problems to analyse the expression power of the proposed interface. One in OpenMP and the other two obtained from this first version by adapting it through a prototype of the proposed interface with versioned Vanilla-TM, that suports transactional memory offered by TinySTM and GCC-TM. The experiments considered three input sizes of the problem, defined according to the nature of each application, identified by: small, medium and large.

Table 1 exemplify the results obtained by presenting the average execution time obtained by running the programs on small, medium and large size entries with 32 threads in the OpenMP execution team.

	Hydra			**Tekoha**		
	OpenMP	*TinySTM*	*GCC-TM*	*OpenMP*	*TinySTM*	*GCC-TM*
hull	2,42	9,15	149,65	2,42	13,27	267,01
norm	0,61	1,28	47,54	0,15	0,63	80,79
outer	0,81	4,53	131,83	1,92	2,24	241,51
sor	4,01	4,14	4,51	0,79	0,9	1,06
thresh	1,55	265,14	502,84	0,87	490,6	1221,75
vecdiff	0,86	2,5	221,55	0,19	1,43	399,66

Table 1. Case performance: Input large, with 32 threads at execution time (time in seconds).

In the tables presented, the cells marked in red correspond tothe experiments whose samples did not adhere to a normal curve.That is, these averages do not represent samples whose distribution has been shown to adhere to a normal distribution. In this case, theaverage times are presented in an illustrative way, but were not used in the performance comparison between the tools.

In the second stage of experimentation, two programs originally developed for performance analysis were implemented with the proposed interface and their performance results were evaluated. One of these programs, bayes, belongs to the benchmark STAMP. The second program was developed as a benchmark for OpenMP. The objective of this second stage of experimentation is to position the proposed solution in relation to its applicability in problems conceived within the context of programming with transactional memory and problems designed to be applied over pure OpenMP. Table 2 exemplify the results obtained.

	Hydra						Tekoha					
	2	4	8	16	32	64	2	4	8	16	32	64
						Small						
kmeans	19,40	10,10	5,29	2,92	1,86	1,62	10,78	7,40	5,05	3,97	3,96	3,89
kmeans-VAN	32,91	38,65	83,84	167,36	401,66	534,04	37,24	50,40	117,27	223,94	459,53	918,57
bayes	10,60	10,84	12,91	11,04	11,86	11,42	5,45	5,76	6,73	5,97	5,29	5,71
bayes-VAN	6,76	3,66	5,70	6,41	3,77	4,10	3,55	2,24	2,81	3,40	2,96	2,17
						Medium						
kmeans	37,12	19,34	10,05	5,56	3,51	3,03	20,70	14,38	10,31	8,16	7,76	7,54
kmeans-VAN	23,92	88,25	152,68	372,36	846,79	1011,78	56,19	113,27	224,95	480,41	930,51	1842,23
bayes	14,12	18,21	12,36	15,67	19,99	16,46	7,33	8,73	9,46	7,04	7,55	8,46
bayes-VAN	8,57	5,05	16,38	9,98	9,12	8,93	3,60	4,77	7,66	4,72	3,23	7,02
						Large						
kmeans	75,16	39,33	20,67	11,10	6,89	6,00	40,86	29,29	21,72	17,30	15,97	15,12
kmeans-VAN	147,07	221,28	335,41	761,71	1677,89	2152,70	126,21	247,09	485,72	1031,31	1872,23	3570,21
bayes	14,50	15,99	13,34	14,18	14,35	13,68	8,30	6,63	8,80	8,80	7,43	7,55
bayes-VAN	7,41	7,64	4,64	8,34	4,50	5,03	5,28	4,46	2,18	3,93	2,70	2,90

Table 2. Comparison: bayes and kmeans

In this experiment, it was observed that the samples adhered to a normal distribution in a greater number of cases. In the Tekoha machine, perhaps because this machine offers greater processing power, the management cost was not compensated by the size of the problems explored. This statement is presented considering cases where the cost of the processing problem is smaller than the exploited parallelism capacity, the execution time in Tekoha is higher. From the analysis of the samples, there is an indication that the bayes-VAN version shows a performance gain compared to its original version.

4. CONCLUSION

This work proposed an extension for OpenMP to support TM. The main result was the incorporation of the transactional memory mechanism in OpenMP and its validation, carried out with the support of a prototype developed on tools offering support to TM, TinySTM and GCC-TM. A performance evaluation was executed implementing benchmarks programs for TM (Bayes) and OpenMP (kmeans) applying the proposed interface. The results allowed to analyse the cost added by the proposed interface in comparison with pure OpenMP and TinySTM. They allowed us to suppose a stability in the overhead using different number of threads regardless of the employed problem size. They also indicated that the proposed interface presents a low overhead in relation to the direct use of TinySTM over OpenMP.

5. BIBLIOGRAPHIC REFERENCES

HARRIS, T., LARUS, J., and RAJWAR, R. **Transactional Memory, 2nd Edition (2nd ed.)**. USA: Morgan and Claypool Publishers, 2010.

HARRIS, T., MARLOW, S., PEYTON-JONES, S., and HERLIHY, M. Composable Memory Transactions. **Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Association for Computing Machinery**, New York, p.48–60, 2005.

SRIVATSA, S. K., and KUMAR, Ch. R. Reconfigurable Frame Work for Chipmultiprocessors and its Application in Multithreaded Environment. **International Journal on Information Sciences and Computing**, p.41–48, 2012.

WAMHOFF, J. T., RIEGEL, T., FETZER, C., and FELBER, P. RobuSTM: A robust software transactional memory. **Symp on Self-Stabilizing Systems**, Springer, p.388–404, 2010.

BAEK, W., MINH, C. C., TRAUTMANN, M., KOZYRAKIS, C., and OLUKOTUN, K. The OpenTM Transactional Application Programming Interface. **Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques (PACT '07)**. **IEEE Computer Society**, Washington, DC, USA, 376–387, 2007.

WONG, M., AYGUADÉ, E., GOTTSCHLICK, J., LUCHANGCO, V., SUPINSKI, B. R., and BIHARI, B. Towards Transactional Memory for OpenMP. In DeROSE, L., SUPINSKI, B. R., OLIVIER, S. L., CHAPMAN, M. B., and S.MULLER, M. **Using and Improving OpenMP for Devices, Tasks, and More**. Springer International Publishing, 2014. Cham, p.130–145.

WONG, M., BIHARI, B. L., SUPINSKI, B. R., WU, P., MICHAEL, M., LIU, Y., and CHEN, W. A Case for Including Transactions in OpenMP. In SATO, M.; HANAWA, T.; S. MULLER, M.; CHAPMAN, B. M.; and SUPINSKI, B. R. **Beyond Loop Level Parallelism in OpenMP: Accelerators, Tasking and More**. Springer Berlin Heidelberg, 2010. p149–160.

WILSON, G. V. Assessing the usability of parallel programming systems: The-Cowichan problems. In DECKER, K. M., and REHMANN, R. M., **Programming Environments for Massively Parallel Distributed Systems**. Birkhäuser Basel, 1994. p. 183-193.

MILOVANOVIC, M., FERRER, R., GAJINOV, V., UNSAL, O. S., CRISTAL, A., AYGUADÉ, E., and VALERO, M. Nebelung: Execution Environment for Transactional OpenMP. **International Journal of Parallel Programming**, Springer, p.326–346, 2008.

Free Software Foundation, Inc 2020. The GNU Transactional Memory Library. https://gcc.gnu.org/onlinedocs/libitm.pdf.