Thread-Based Competitive Or-Parallelism

Paulo Moura
Dep. of Computer Science
University of Beira Interior, Portugal
pmoura@di.ubi.pt
Center for Research in Advanced Computing Systems
INESC–Porto, Portugal

Ricardo Rocha
Dep. of Computer Science
University of Porto, Portugal
ricroc@dcc.fc.up.pt
Center for Research in Advanced Computing Systems
INESC–Porto, Portugal

Sara C. Madeira
Dep. of Computer Science
University of Beira Interior, Portugal
smadeira@di.ubi.pt
Knowledge Discovery and Bioinformatics Group
INESC–ID, Portugal

1 Main Concept

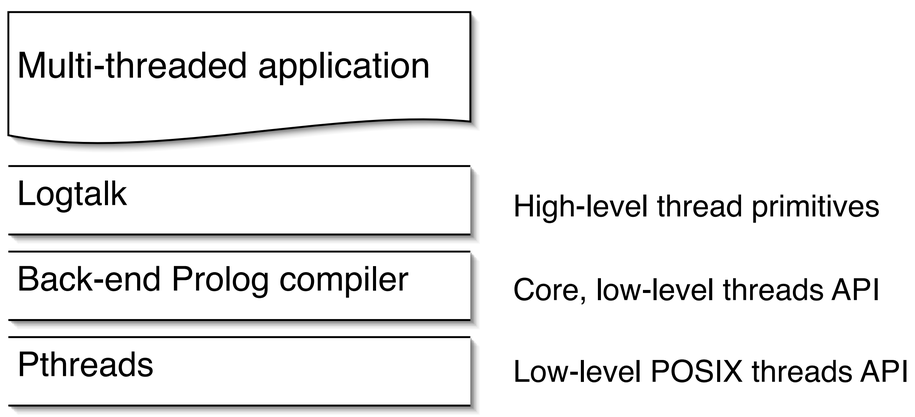
The concept of ***thread-based competitive or-parallelism*** combines the original idea of *competitive or-parallelism* with *committed-choice nondeterminism* and *speculative threading*. In thread-based competitive or-parallelism, an explicit disjunction of subgoals is interpreted as a set of concurrent alternatives, each running in its own thread. The subgoals compete for providing an answer and the first successful subgoal leads to the termination of the remaining ones. Thread-based competitive or-parallelism is implemented in Logtalk, an object-oriented logic programming language that can use most Prolog implementations as a back-end compiler.

2 Background

Problem: Plenty of high quality research on logic programming parallel systems. However, most parallel systems described in the literature are no longer available, due to the complexity of maintaining and porting their implementations.

Our approach: Implement high-level parallelism programming constructs in Logtalk using the core Prolog multi-threading support found in several compilers, which are available for most operating-systems. Logtalk provides a single built-in predicate, `threaded/1`, supporting both independent and-parallelism and competitive or-parallelism.

Motivation: Many applications need simple, high-level programming constructs for running some tasks concurrently. Programmers prefer to avoid low-level, error prone implementation details such as creating or synchronizing threads. Powerful multi-core personal computing systems are increasingly available.



3 An Example: The Generalized Water Jug Problem

Table 1 shows the running times, in seconds, when 5-liter and 9-liter jugs were used to measure from 1 to 14 liters of water. It allows us to compare the running times of single-threaded depth-first (DF), breadth-first (BF), and hill-climbing (HC) search strategies with the competitive or-parallelism (COP) multi-threaded call where one thread is used for each individual search strategy. The results show the average of thirty runs. The fastest method for each measure is highlighted. The last column shows the number of steps of the solution found by the competitive or-parallelism call. The maximum solution length was set to 14 steps for all strategies.

Our experimental setup used Logtalk 2.33.0 with SWI-Prolog 5.6.59 64 bits as the back-end compiler on an Intel-based computer with four cores running Fedora Core 8 64 bits.

Table 1. Measuring from 1 to 14 liters with 5-liter and 9-liter jugs.

Liters	DF	HC	BF	COP	Overhead	Steps
1	26.373951	0.020089	**0.007044**	0.011005	0.003961	5
2	26.596118	12.907172	**8.036822**	8.324970	0.288148	11
3	20.522287	**0.000788**	1.412355	0.009158	0.008370	9
4	20.081001	**0.000241**	0.001437	0.002624	0.002383	3
5	**0.000040**	0.000240	0.000484	0.000907	0.000867	2
6	3.020864	0.216004	**0.064097**	0.098883	0.034786	7
7	3.048878	**0.001188**	68.249278	0.008507	0.007319	13
8	2.176739	**0.000598**	0.127328	0.007720	0.007122	7
9	2.096855	**0.000142**	0.000255	0.003799	0.003657	2
10	**0.000067**	0.009916	0.004774	0.001326	0.001295	4
11	**0.346695**	5.139203	0.587316	0.404988	0.058293	9
12	14.647219	**0.002118**	10.987607	0.010785	0.008667	14
13	0.880068	0.019464	**0.014308**	0.029652	0.015344	5
14	0.240348	0.003415	**0.002391**	0.010367	0.007976	4

4 Pros and Cons

Pros: Simple to use. Simple semantics. Simple and portable implementation. Most useful when combined with other forms of parallelism. Potential performance boost when combined with tabling. Concept orthogonal to the object-oriented features of Logtalk.

Cons: Inherent multi-threading overhead. Thread cancellation issues. Requires a minimum of a processing core per thread for effective results. No load-balancing mechanism in the current implementation (other than the native operating-system mechanism).

References

Ertel, W.: *Performance Analysis of Competitive Or-Parallel Theorem Proving*. Technical report fki-162-91, Technische Universitat Munchen (1991)

Shapiro, E.: *The Family of Concurrent Logic Programming Languages*. ACM Computing Surveys 21(3) (1989) 413–510

González, A.: *Speculative Threading: Creating New Methods of Thread-Level Parallelization*. Technology@Intel Magazine (2005)

Moura, P.: *ISO/IEC DTR 13211–5:2007 Prolog Multi-threading Support* Available from http://logtalk.org/plstd/threads.pdf

Gupta, G., Pontelli, E., Ali, K., Carlsson, M., Hermenegildo, M.V.: *Parallel Execution of Prolog Programs: A Survey*. ACM Transactions on Programming Languages and Systems 23(4) (2001) 472–602

Moura, P.: *Logtalk – Design of an Object-Oriented Logic Programming Language*. PhD thesis, Dep. of Computer Science, University of Beira Interior (2003)

Ali, K., Karlsson, R.: *The Muse Approach to OR-Parallel Prolog*. International Journal of Parallel Programming 19(2) (1990) 129–162

Acknowledgments

This work has been partially supported by the FCT research projects STAMPA (PTDC/EIA/67738/2006) and MOGGY (PTDC/EIA/70830/2006). We are grateful to Jan Wielemaker, Vítor Costa, Terrance Swift, and Rui Marques for their groundwork implementing Prolog multi-threading core support and for helpful discussions on the subject of this work.