Improving Scalability of Task-Based Programs

Iulian Brumar, Marc Casas, Miquel Moretó
Barcelona Supercomputing Center
ibrumar@bsc.es, marc.casas@bsc.es, miquel.moreto@bsc.es

Abstract-
In a multi-core era, parallel programming allows further performance improvements, but with an important programmability cost. We envision that the best approach to parallel programming that can exceed the programability, parallelism, power, memory and reliability walls in Computer Architecture is a run-time approach.
Many traditional computer architecture concepts can be revisited and applied at the runtime layer [4][5] in a completely transparent way to the programmer. The goal of this work is taking the computer architecture value prediction and data-prefetching concepts inside a runtime environment like OmpSs.

I.INTRODUCTION

The main objective of this work is researching if *Value Locality* exists in state of the art OmpSs programs and if we can use it in order to obtain better execution times.

Value locality is the property of a static instruction to produce the same output given the same input. If, let us say, a hardware *sum* instruction it is executed twice in a loop, and both times it gets exactly the same inputs, for its second execution we already know that it will generate the same output. However in hardware load instructions, if the input is the same -the address- we are not sure if it will produce the same result. In this case we can only speculate, but even so it has been shown that in many cases, static loads with same input produce the same output [1].

By using this knowledge, we can build a predictor that will skip those instructions that can be well predicted and feed the depending instructions earlier with the predicted output.

In this work we take this concept to a new level for OmpSs tasks and we can distinguish two sub-objectives:

1) Analyze OmpSs benchmarks predictability. We cannot prove that value locality will lead to performance improvement for all possible programs, but we can at least focus on state of the art applications that have been ported to the OmpSs programming model and see how can the value locality concept be extended to our context.

2) Find the ideal speedup using a value locality predictor. This second objective it is a consequence of the previous one. In the cases where value locality exists, what performance improvement can be achieved? We will answer this question using simulation tools.

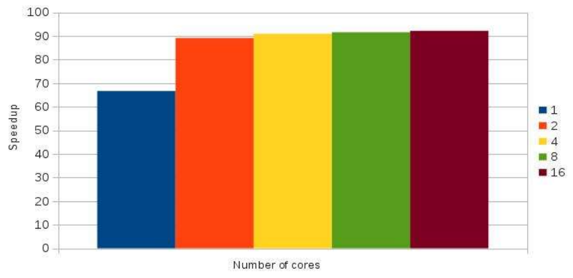
Notice that this is a best case approach in order to discover the limits of the predictability we can have. Also we have to mention that this work was performed with fine grained tasks.

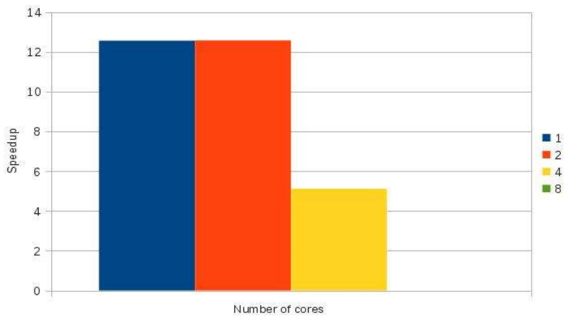
II.RELATED WORK

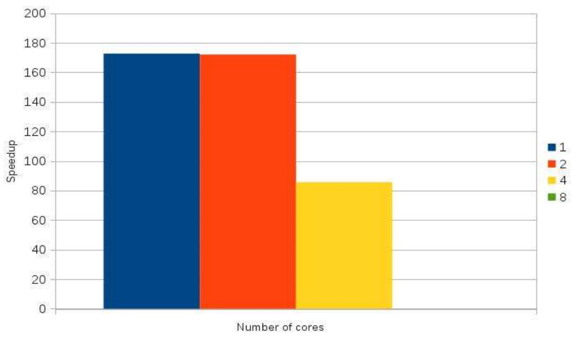
Since the first moments of computer architecture, it has been seen that the dependencies between instructions were a big wall against *Instruction Level Parallelism* (*ILP*). A good example of instruction level parallelism is the pipelined processor, which is made of several hardware slots, each one with a specific function. If there are two slots in our processor, namely A and B, an instruction must fulfill both stages in order to complete its execution. We call this an instance of *ILP* because the processor can have two instructions running at the same time. If we hadn't pipelined the processor every instruction would have executed in time time(A) + time(B) but this technique allows us to execute a instruction in time max(time(A), time(B)).

The problem is that the instruction in the first stage (A) might need the result produced by the oldest instruction in (B). In this case the newest instruction will spend one more cycle in stage A and this is a conflict caused by a Read After Write RAW dependency. Even so, back to the 90's, the architects came with a solution [1]. The idea was to continue the execution of the instructions affected by the conflict speculatively. In our example it means that the instruction in stage A can complete the process in this stage speculating the result of the instruction in stage B, and check if the supposition was correct in the next stage.

Now a very good question would be: How can processors predict well the results of hardware instructions? That issue has been explored in the papers of Lipasti [2] and Sazeides [1] which form the motivational base of this work. In the first one the predictor is implemented in two different processor architectures (the out of order PowerPC and the in order Alpha), while the second article gives a more theoretical approach to the issue explaining computational predictors (explained in more detail in [3]) and context based predictors.







III.SOME RESULTS

Figure 1 shows the performance improvement for the Jacobi, Blackscholes and CheckSparseLU benchmarks.

Fig. 1. Performance improvement of Jacobi, Blackscholes and CheckSparseLU.

As we were mentioning in the introduction, those results are obtained using very small task granularities. Additionally, in those three benchmarks, for the same input, the same output is guaranteed to be produced (unlike some programs that don't specify all the data used in their dependencies). For more details on the executions see Table 1. Those speedups are obtained via simulation with TaskSim.

TABLE I

BENCHMARKS CHARACTERISTICS

	Jacobi	Blackscholes	CheckSparseLU
Num. Tasks	64	1024	5000
Bytes/Task	~512	~256	~256
Predicted Tasks	38	899	4800

IV.CONCLUSIONS AND FUTURE WORK

Although huge performance improvements can be achieved using value prediction, we have managed to get these results only at very fine grained levels of parallelism. As part of the same project we have developed a value predictor integrated in the OmpSs runtime together with recovery schemes and data prefetching techniques in case of missprediction.

ACKNOWLEDGMENT

This work has been partially supported by the Spanish Ministry of Science and Innovation under grant TIN2015-65316-P, the HiPEAC Network of Excellence, and by the European Research Council under the European Union's 7th FP, RoMoL ERC Advanaced Grant Agreement number 321253. M. Moreto has been partially supported by the Ministry of Economy and Competitiveness under Juan de la Cierva postdoctoral fellowship number JCI-2012-15047. M. Casas is supported by the Secretary for Universities and Research of the Ministry of Economy and Knowledge of the Government of Catalonia and the Cofund programme of the Marie Curie Actions of the 7th R&D Framework Programme of the European Union (Contract 2013 BP_B 00243).

REFERENCES

[1] SAZEIDES, Yiannakis; SMITH, James E. "The predictability of data values". En *Microarchitecture*, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on. IEEE, 1997. p. 248-258.

[2] LIPASTI, Mikko H.; WILKERSON, Christopher B.; SHEN, John Paul. "Value locality and load value prediction". ACM SIGOPS *Operating Systems Review*, 1996, vol. 30, no 5, p. 138-147.

[3] LIPASTI, Mikko H.; SHEN, John Paul. *Exceeding the dataflow limit via value prediction*. En *Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture. IEEE Computer Society*, 1996. p. 226-237.

[4] Marc Casas, Miquel Moreto, Lluc Alvarez, Emilio Castillo, Dimitrios Chasapis, Timothy Hayes, Luc Jaulmes, Oscar Palomar, Osman Unsal, Adrian Cristal, Eduard Ayguade, Jesus Labarta, and Mateo Valero. *Runtime-aware architectures*. In *Euro-Par*, pages 16–27. 2015.

[5] Mateo Valero, Miquel Moreto, Marc Casas, Eduard Ayguade, and Jesus Labarta. "Runtime-aware architectures: A first approach." *International Journal on Supercomputing Frontiers and Innovations*, 1(1):29–44, June 2014.