Exploring some multicore research opportunities. A first attempt.

Ciprian Radu*, Horia Calborean*, Adrian Florea*, Árpád Gellért*, and Lucian Vinţan*

* *ACAPS (http://acaps.ulbsibiu.ro/research.php), Lucian Blaga University, Emil Cioran 4, 550025 Sibiu, Romania*

ABSTRACT

Multicore architectures are currently the most common solution for further increasing the processing performance since the methods for exploiting the Instruction Level Parallelism (ILP) have reached a certain saturation point. However, we believe that multicores should still consider the benefits provided by ILP exploitation mechanisms. In this respect, we consider integrating techniques like Value Prediction and Dynamic Instruction Reuse into Chip Multiprocessor architectures. Also, we present the need for evaluating multicore architectures by Automatic Design Space Exploration.

KEYWORDS: Multicore; Value Prediction; Dynamic Instruction Reuse; Automatic Design Space Exploration

1 Introduction

Nowadays, computer architects are confronted with the power wall: the clock frequency can no longer be increased because of the dynamic power consumption ($P_d = kCV^2f$) and thermal dissipation. The performance can be grown only by architectural innovations, and the essential challenge is to determine the optimal compromise between the processing performance and the architectural complexity. Since the methods for exploiting the Instruction Level Parallelism have reached a certain saturation point, the most common solution for further increasing the performance is given by multicores because they provide a better performance/Watt ratio than monoprocessors. However, multiprocessors involve parallel programming, and writing a parallel program is more difficult than writing a sequential program. Therefore, designing multicore systems must account for the productivity of the programming process and for the performance of the software application too.

[1]E-mail: {ciprian.radu,horia.calborean,adrian.florea,arpad.gellert,lucian.vintan}@ulbsibiu.ro

2 Research challenges

Currently, 99% of the software programs are sequentially written and as Amdahl's law shows, the sequential fraction of a program can dramatically limit the potential speedup provided by multicores. Multicores should not neglect the ILP and their research should consider the synergistic exploitation of different kinds of parallelism (pipeline, ILP, TLP, tasks, data, etc.). Value Prediction and Dynamic Instruction Reuse techniques were developed in order to reduce the data-flow wall and, particularly, the program's critical execution path. In [GFV09] we have developed a mechanism of selective value anticipation dedicated to high latency instructions, which includes a reuse scheme for MULs and DIVs, and a value predictor for critical LOADs (miss in L2-cache). On a superscalar architecture, our results show an IPC increase by 3.5% on the SPEC 2000 integer benchmarks and an increase by 23.6% on the floating point SPEC 2000 benchmarks. The Energy Delay Product decreases with 6.2% in the case of the integer benchmarks and respectively with 34.5% for the floating point benchmarks. We have also successfully applied those mechanisms into a Simultaneous Multithreading Architecture.

2.1 Using Value Prediction in Multicores

In a multicore architecture a simple implementation of value prediction is incorrect because it can violate the memory consistency model [MSC+01]. The next example shows that even correctly predicting a LOAD can generate incorrect program execution. Processor P1

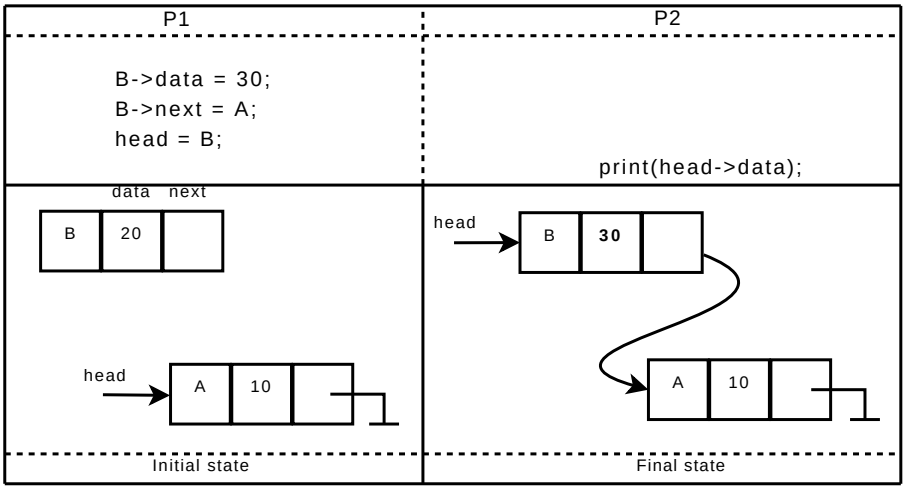

Figure 1: Simple value prediction could violate memory consistency

changes the value from node B from 20 to 30 and before attaching it to the head of the list (initially made only by node A). Processor P2 prints the value from the head of the list. If P2 would use value prediction, it could correctly predict that the head of the list will be node B, and then it could print value 20 which is a violation of sequential consistency. Two detection mechanisms for memory consistency violation are proposed. The first is based on addresses: a processor must detect when another processor writes to an address that was speculatively read. The other detection mechanism is rather based on values: each speculative load will

wait until its operands become non-speculative and then the load will be performed once again (this time non-speculative).

2.2 Performing Dynamic Instruction Reuse in Multicores

We now consider a multicore architecture with DIR used in each core. The following program is executed, considering a shared memory programming model. This program repre-

```
// let k be a strictly positive integer number and let V be a shared variable
int i = 0;
while (i < k) {
      bool new_value_produced = false;
      if (pthread_self() == 0) {
            if (i > 0) {
                  // wait until the rest of the threads consume the previous value
                  while (!previous_value_consumed);
            }
            V = i; // thread 0 produces a new value for each loop iteration
            new_value_produced = true;
      } else {
            // wait until thread 0 produces the new value
            while (!new_value_produced);
            printf("%d\n", V); // consume the new value
            // this thread will have to notify that it consumed the value...
      }
      i++;
}
```

Figure 2: A one-producer-many-consumer scenario

sents a one-producer-many-consumer scenario: at each iteration, the first processor writes a new value into the shared variable V, and the rest of the processors read the new value of V. From the DIR point of view, for the processors which consume the value of V, we can see that the load instruction which corresponds to reading the new V value can be stored in the Reuse Buffer. This way, for the next iterations, loading the value of V can be done faster. However, it is obvious that the value of this load instruction could not be reused because each time V will hold a new value. The invalidation mechanism for the Reuse Buffer must be implemented globally because both local and remote stores have to be considered. The solution would be to benefit from the invalidation messages sent at the level of the processor's cache. Cache coherence can help at keeping the Reuse Buffer data correct.

2.3 Automatic Design Space Exploration

Simulating multicore architectures in order to find the optimum configuration proves to be extremely time consuming because of the enormous design space of the researched applications, compiler's parameters and architectures. Parallel benchmarks have to be used and the simulation environment must provide support for multicores. Below it is presented a simulation result from our previous work with UNISIM(`http://unisim.org`): we have evaluated the evolution of the global IPC of a dual-core Symmetric Multiprocessor. It can be observed that besides the parameters specific to a single processor architecture, we have to take into account more parameters, the ones that are multicore specific. In the more general case, we should consider that our further developed multicore architecture can contain: different types of processing elements, different types of interconnection networks, multiple memory consistency models, some compiler optimization techniques etc. Therefore, an important challenge in the field of CMPs is Automatic Design Space Exploration (ADSE): developing

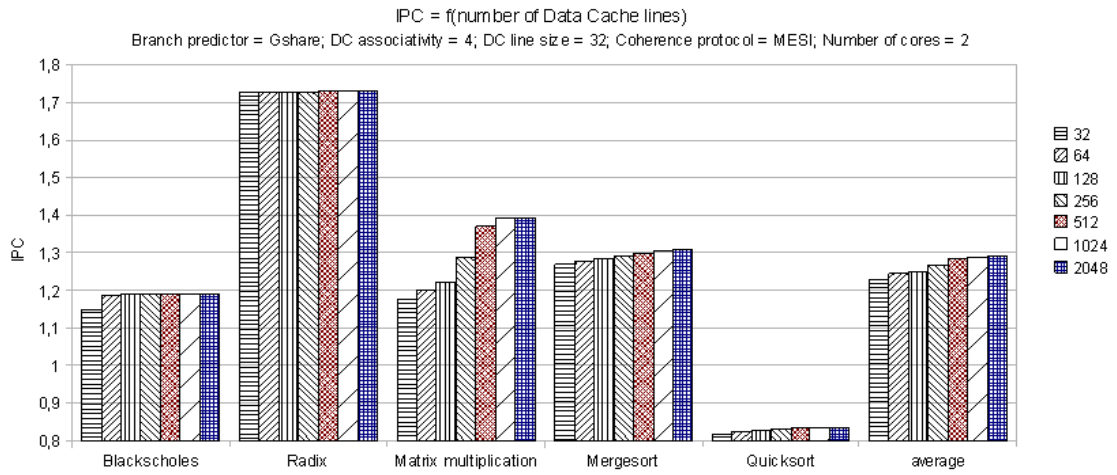

Figure 3: IPC as a function of the number of lines of the data cache from a dual-core SMP

optimized heuristic research methods for the huge space of the parameters of applications, compiler and architectures. ADSE has the purpose to automatically optimize the architectural parameters by taking into account multiple objectives (IPC, power consumption, integration area etc.). Obviously the search in this huge design space cannot mean considering every possible configuration. Exhaustive techniques must be replaced by heuristic-based search methods. Local search algorithms (e.g.: hill climbing, simulated annealing), genetic algorithms, ant colony algorithms, etc. could prove to be useful. Other advanced machine learning techniques can further help by reducing the number of simulations required to find a near optimal architectural configuration.

3 Conclusions

We believe that the already developed mechanisms for exploiting Instruction Level Parallelism must be used in multicore architectures. Techniques like Value Prediction and Dynamic Instruction Reuse can help in increasing the performance over the sequential parts of the parallel programs. However, such techniques have to be adapted to the more general context provided by parallel programming. Also, developing a multi-criteria ADSE would help in optimizing the process of determining the best architectural configuration.

References

[GFV09] Arpad Gellert, Adrian Florea, and Lucian Vintan. Exploiting selective instruction reuse and value prediction in a superscalar architecture. *Journal of Systems Architecture*, 55(3):188–195, 2009.

[MSC+01] Milo M. K. Martin, Daniel J. Sorin, Harold W. Cain, Mark D. Hill, and Mikko H. Lipasti. Correctly implementing value prediction in microprocessors that support multithreading or multiprocessing. In *MICRO 34: Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture*, pages 328–337, Washington, DC, USA, 2001. IEEE Computer Society.