CHAPTER 4

INSTRUCTION-BASED DYNAMIC CLUSTER ASSIGNMENT MECHANISMS

In this chapter we propose and analyze several dynamic cluster assignment schemes that will be referred to as the *instruction-based steering schemes*. These schemes work at a finer granularity than the slice-based schemes studied in the previous chapter, because they assign instruction per instruction to clusters instead of considering instruction groups. In both cases, the main steering decisions are based on the dependences through registers but, unlike the slice-based schemes, the instruction-based ones only consider the immediate dependences of every dynamic instruction with its predecessors in the data dependence graph.

Our main contribution is the Advanced RMB scheme, which assigns instructions following primary and secondary criteria. The primary criterion selects clusters requiring the fewest communications to read the source registers. If more than one cluster is selected, the secondary criterion chooses the least loaded one. However, the algorithm ignores the primary criterion in case the workload imbalance exceeds a given threshold. We also present two improvements to this algorithm. The first one is the Priority RMB scheme, that slightly modifies the primary criterion: when any source register is unavailable it gives priority to the producer cluster of the unavailable source register. The second one is the Accurate Rebalancing Priority RMB scheme, that slightly modifies the action to be taken when the workload imbalance exceeds the threshold: instead of totally ignoring the primary criterion, it just excludes the overloaded clusters prior to applying the two criteria.

In this chapter, our instruction-based schemes are evaluated in the context of our Reference Cluster Architecture (see Chapter 2), with two and four clusters, and they are compared to the best previously proposed schemes. It is shown that the proposed Accurate Rebalancing Priority RMB scheme significantly outperforms the best previously proposed schemes, since it achieves the best trade-off between communications and workload balance.

4.1 Communication and Workload Balance

In the previous chapter we described the two main goals of a cluster assignment mechanism: minimizing the penalties of inter-cluster communications and maximizing the workload balance. In this section it is described how these two important issues are addressed by the instruction-based partitioning schemes that are proposed.

4.1.1 Communication

Inter-cluster communication penalties may occur if two dependent instructions are steered to different clusters, and they belong to the critical path of execution. Since determining the critical path is a very complex task in the context of a dynamically scheduled processor [107], all the known approaches use some heuristics that approximate this objective. Most of the presented schemes try to simply minimize the number of inter-cluster communications.

In the context of an architecture with distributed register files, like the Reference Cluster Architecture, the instruction results are by default stored only in the register file of the cluster where the instruction executes. Whenever a communication is required to read a source operand, the dispatch logic generates a copy instruction, allocates a new physical register to the same logical register, and updates the map table to show its multiple mappings with the valid bit set. The valid bit associated to each field of the register map table (see chapter 2) indicates whether a source register may be directly read in the corresponding cluster without requiring a communication. The steering schemes proposed in this chapter use this information to minimize communications, hence we refer to them as *register mapping based* (RMB).

Note also that we only analyze steering schemes for architectures with a distributed register file. On some other architectures, the register file is replicated in all clusters, and all copies are kept consistent by broadcasting every result to all clusters (e.g. the Alpha 21264 [53]). In such architectures, the inter-cluster communication latency is hidden if the value has already been produced and broadcast by the time it is needed by the consumer. Thus, the task of minimizing communication penalties is slightly different, since it must take into account only the source operands that are not yet broadcast. While this architectural approach simplifies the steering mechanism, it shifts the complexity into the register file and the bypass network, and will not be analyzed further since it falls beyond the scope of our work.

4.1.2 Workload Balance

In section 3.1 the problem of the workload imbalance among clusters was described as the situation where an instruction is delayed by the lack of available functional units in its cluster, even though there exists idle functional units in other clusters. Since workload imbalance may potentially degrade performance, a major goal of the steering logic is to prevent it from happening. As mentioned in section 3.1.1, obtaining an optimal partitioning that minimizes the delays of critical instructions caused by the workload imbalance is a hard problem, and all the existing algorithms, as well as those proposed here, resort to heuristics to address it.

There are many alternatives to determine at run-time the individual workloads of the clusters and their relative imbalance, because there is not a unique definition. In section 3.1.2 the workload imbalance was reported in terms of metric I2, which was defined as the difference in number of ready instructions between the two clusters. However, such a definition is not suitable for an architecture that may have more than two clusters, like our Reference Cluster Architecture. Therefore, in this chapter it is replaced by a new definition, though similar in concept. From the description of the workload imbalance in the previous paragraph, we intuitively define the workload imbalance at a given instant of time as the total number of ready instructions that cannot issue, but could have issued in other clusters since they have idle functional units. We will refer to this figure as metric NREADY, and it is used to report the workload imbalance in our experiments. To be more precise, if we count for each cluster the difference between the number of ready instructions and its issue width; next we sum separately the positive and the negative differences; then NREADY is defined as the minimum between the absolute values of these two sums.

In section 3.1.2, we found that the steering decisions were best guided by another metric, which was mainly influenced by the number of instructions dispatched to each cluster (metric I1). Unfortunately, as it was defined, I1 is not suitable for an architecture that may have more than two clusters, like our Reference Cluster Architecture. Therefore, in this chapter it is replaced by a new definition, though similar in concept, that will be referred to as metric DCOUNT. We tried also with metrics based on the number of instructions present in the issue queues, but DCOUNT was found to give the best performance. From a conceptual standpoint, DCOUNT may be defined as the maximum of the absolute deviations of the accumulated number of dispatched instructions per cluster. Despite this apparently complex definition, it may be implemented with reasonably low complex hardware: the processor has a signed counter for each of the N clusters that measures its workload. Its value is initially zero, and it is updated in the following way: for every instruction dispatched to a cluster, the corresponding counter in that cluster is increased by N-1, while the other N-1 counters are decreased by 1 (i.e. the sum of the counters is kept always zero). Therefore, the value stored in the counter of a given cluster is N times the difference between the total number of instructions dispatched to that cluster and the average number of instructions dispatched per cluster (the deviation). The workload imbalance is calculated as the maximum absolute value of the workload counters. Note also that in the case of two clusters, DCOUNT is equivalent to I1, and a single counter will suffice. On a branch misprediction, the pipeline is flushed, and the precise state of the workload counters needs to be recovered. However, we found that just clearing all counters performs quite well, and requires much less complex hardware.

The NREADY figure matches more exactly our definition of workload balance. However, when it is used by the steering logic, the actions taken to compensate a workload imbalance (sending instructions to the least loaded cluster) may not update immediately the NREADY figure, if some of the steered instructions are not ready. When this occurs, the corrective action may result disproportionate, and cause an imbalance in another direction or some unnecessary inter-cluster communications. This does not happen with the DCOUNT figure, since it varies instantly and in proportion to the steering decisions, which allows the steering logic to gauge more accurately the actions to compensate a workload imbalance. Thus, the steering logic uses the DCOUNT figure to determine balancing actions and we use the NREADY figure to measure and report workload balance.

4.2 Instruction-Based Cluster Assignment Schemes

This section presents and evaluates several instruction-based dynamic cluster assignment schemes. This kind of algorithms are finer-grain than slice-based ones, as they work instruction by instruction instead of considering instruction groups. On the one hand, they ignore the relationship with other instructions that cooperate in the calculation of the same load address or branch condition unless they are directly dependent, but the finer granularity make them more flexible to assign instructions to clusters.

All the proposed RMB steering algorithms try to minimize the number of communications required by an instruction by choosing a cluster where the highest number of its source operands are currently mapped. Since this criterion is included in all schemes, we outline its rules here once, to improve readability:

- If the instruction has no register source operands, all the clusters are considered as potential clusters to dispatch the instruction.
- If the instruction has one register source operand, only those clusters where it is mapped are considered.
- If the instruction has two register source operands and there is at least one cluster where both are mapped, those clusters that have both operands mapped are considered; otherwise those clusters where one of the registers is mapped are considered.

Note that this steering criterion may not deliver always a unique cluster selection, so another secondary criterion must be considered to further refine it.

4.2.1 Experimental Framework

The experiments in this chapter evaluate the instruction-based schemes assuming the Reference Clustered Architecture and all the experimental setup described in chapter 2, including the simulator, benchmarks and architectural parameters. In this section, for simplicity, we assume a configuration with four clusters and the SpecInt95 benchmark suite [97], but in section 4.3.5 we study configurations with two and four clusters and they are evaluated using both the SpecInt95 and the Mediabench [56, 62] benchmark suites.

The IPC of the clustered architecture is compared to the IPC of a centralized architecture because it is an upper bound reference that helps to see how much IPC they are trading-off for clock speed and power savings. The following subsections describe the steering schemes along with several evaluations that motivate their design. A more complete evaluation including comparison with other previously proposed schemes in the literature will be conducted in section 4.3.

4.2.2 Modulo Steering

This is actually not a dependence-based scheme, but we introduce it here for comparison purposes. This is a naive scheme that sends alternatively one instruction to each cluster. It is simple and very effective regarding workload balance, but may result in a high communication overhead because there is no consideration on communication.

4.2.3 Simple RMB Steering

The first RMB steering algorithm we have considered is the Simple RMB scheme. First, it selects the clusters where all or most of the source operands are mapped, as described above, then it chooses one of them randomly. The algorithm works as follows:

1. Select clusters with highest number of source registers mapped (as described in section 4.2)
2. Choose one of the above selected clusters, randomly

A communication is originated just in the case when one instruction has two source registers and there is not any cluster that has both registers mapped. No consideration of balance is taken, relying on the fact that the random steering used when the communication overhead is the same for several clusters is good enough to keep the balance steady.

4.2.4 Balanced RMB Steering

The Balanced RMB scheme includes some workload balance considerations. Whenever there is no preferred cluster from the point of view of communication overhead, the balance is taken into account and the instruction is sent to the least loaded cluster.

1. Select clusters with highest number of source registers mapped
2. Choose the least loaded among the above selected clusters

This scheme will improve significantly the workload balance while trying to keep the communications to a minimum since the balance is just taken into account whenever several clusters are considered equally good from the communications point of view.

Figure 4-1 shows the IPC for the Modulo, Simple RMB, Balanced RMB, and the upper bound centralized architecture. Figure 4-2 shows the average number of communications per committed instruction, and figure 4-3 shows the average workload imbalance. As expected, the modulo scheme is the best balanced one, since it sends one instruction to each cluster alternatively. However, while this scheme achieves a near optimal workload balance, it shows the worst performance, because the advantage in balance cannot offset the overhead produced by a huge amount of inter-cluster communications (almost 98% of the instructions executed require communications).

In contrast, the Simple RMB and the Balanced RMB schemes perform significantly better due to their much lower communication overhead. The Balanced RMB steering has a better

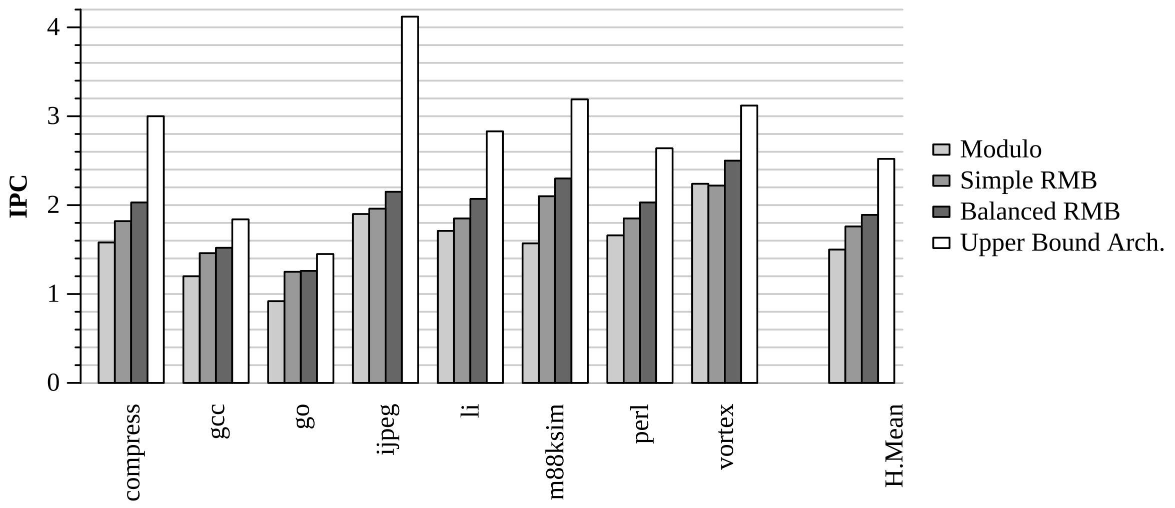

Figure 4-1: Performance of Modulo, Simple RMB and Balanced RMB steering schemes on a four-cluster architecture

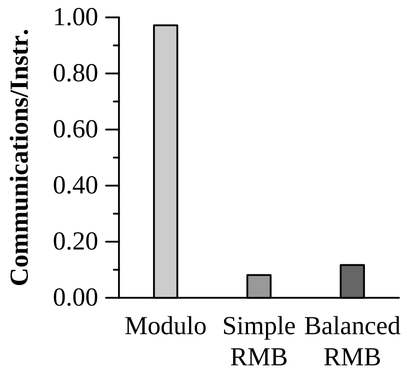

Figure 4-2: Average number of communications per dynamic instruction

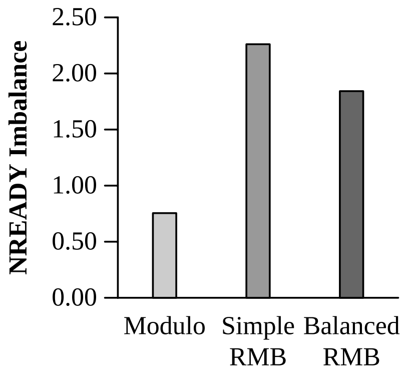

Figure 4-3: Average NREADY workload imbalance

workload balance than the Simple RMB, as shown in figure 4-3, due to the balance considerations it implements, and therefore, it achieves also a better performance.

Overall, the above results show that, while it is important to achieve a good workload balance among the clusters, it is also important to reduce the communications. We can conclude that, among the three algorithms, the Balanced RMB scheme performs the best because it achieves the best trade-off between communication and workload balance.

However, the IPC of the Balanced RMB is still significantly below that of the centralized - upper bound - architecture (figure 4-1). To attempt improving it we have also studied the behavior of the workload balance along the execution of each program. We measured the distribution function of the NREADY metric for each program. This function is depicted in Figure 4-4, only for a particular benchmark (perl) although we found that the shape of the distribution is quite similar for all the benchmarks examined. The graph shows that the most frequent imbalance value is zero, but this is not a surprising result, since the NREADY metric only considers imbalance as the situation where some cluster has more ready instructions than

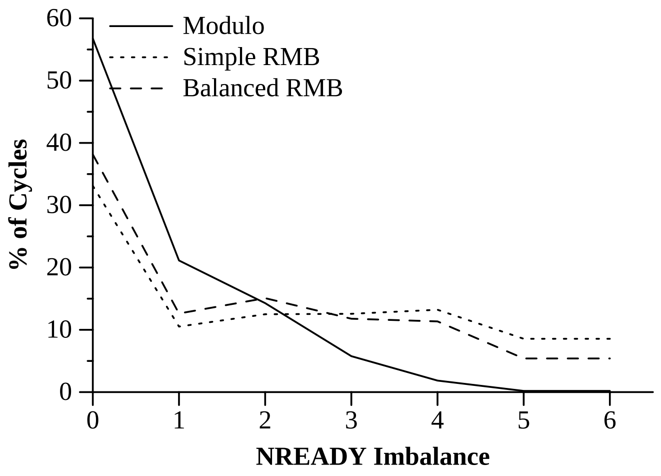

Figure 4-4: Distribution function of the NREADY workload imbalance (perl)

the issue width and another cluster has fewer. The most remarkable feature of this graph is that high imbalance values (NREADY > 2) are not infrequent, which motivates the proposal of a new steering scheme.

4.2.5 Improving the Workload Balance with the Advanced RMB Steering

Figure 4-1 showed that the harmonic mean IPC of the Balanced RMB is 1.89, still below that of the centralized architecture. Since Figure 4-4 showed that high imbalances (NREADY>2) are not infrequent, which suggests that there is still some potential improvement by trying to avoid those strongly imbalanced situations. The Advanced RMB scheme is similar to the Balanced RMB, with a higher emphasis in the workload balance. This scheme checks whether the imbalance has exceeded a given threshold and in this case it ignores the primary criterion, and it just sends the instruction to the cluster that most favours the balance. The algorithm works as follows:

1. Select clusters with highest number of source registers mapped
2. Choose the least loaded among the above selected clusters

Exception: if imbalance is greater than a given threshold, then ignore rule 1

Of course, this approach may decide that an instruction executes in a cluster where none of its operands are mapped, due to the poor workload balance at that moment, and therefore it may increase the number of inter-cluster communications. The threshold determines a trade-off between workload balance and inter-cluster communications. If the threshold is set too high, then we get little improvements on workload balance, but if it is set too low, then the communication overhead offsets the balance improvements. Some experiments have been conducted in order to find out the best threshold that triggers re-balancing. We found the best values to be DCOUNT=16 for 2 clusters, and DCOUNT=32, for 4 clusters.

In Figure 4-5 it is shown the IPC for the Modulo, Balanced RMB and Advanced RMB. Figure 4-6 shows the average number of communications per executed instruction, and figure 4-7 shows the average workload imbalance. It is shown in figure 4-7 that the Advanced RMB scheme achieves the best workload balance. This workload balance improvement is better

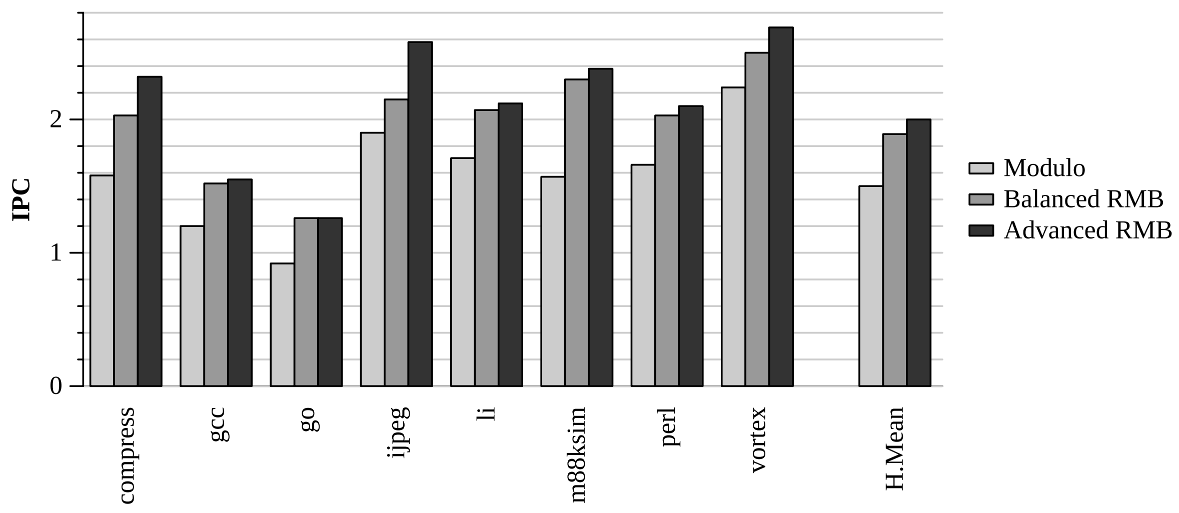

Figure 4-5: IPC of the Advanced RMB vs. Balanced RMB steering schemes on a four-cluster architecture

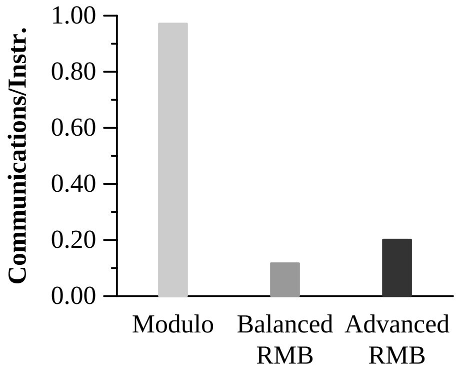

Figure 4-6: Average number of communications per dynamic instruction

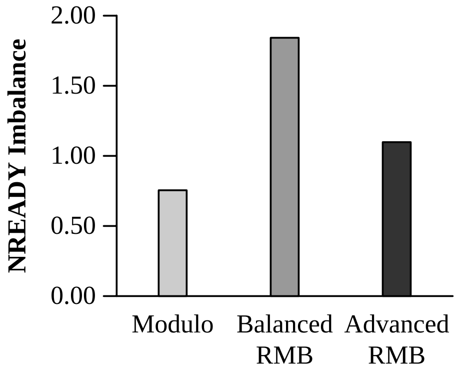

Figure 4-7: Average NREADY workload imbalance

explained in figure 4-8, which depicts its distribution function. It shows that the rebalancing actions produced by the Advanced RMB steering scheme mostly tend to reduce the frequency of situations with a high NREADY imbalance. As shown in figure 4-6, such a balance improvement comes at the cost of an increase in communications because, during rebalancing actions, the dependence criterion is ignored. However, as shown in figure 4-5, the Advanced RMB outperforms the other schemes, because it achieves the best trade-off between communications and workload balance.

4.2.6 Optimizing the Critical Path with the Priority RMB Steering Scheme

One of the main goals of a steering algorithm is to minimize the performance penalty associated to inter-cluster communications. However, not all communication delays result in a loss of performance but only those among instructions in the critical path of execution. Therefore, a more effective approach would be to minimize the number of critical communications.

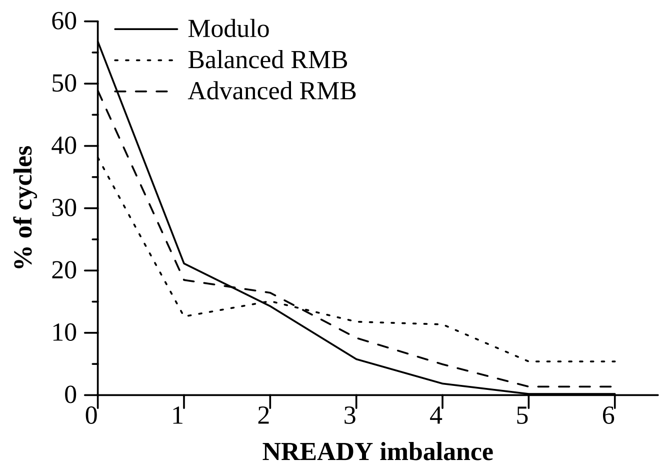

Figure 4-8: Distribution function of the NREADY workload imbalance (perl)

This kind of considerations motivate our Priority RMB scheme, which addresses the relative criticality of the communications within the scope of a single instruction. Note that among the true register dependences of an instruction, only the last operand being produced may be critical. Our approach is based on the observation that if an instruction has two source registers, and only one of them is available, the unavailable operand is more likely to be critical, because it will be produced later. Consequently, the Priority RMB steering scheme is similar to the Advanced RMB, except that it considers the dependences through unavailable operands prior to other dependences. The algorithm works as follows:

```
1.  To minimize communication penalties:
    1.1.  If there is any unavailable operand, choose its producer cluster
    1.2.  Else, select clusters with highest number of source registers mapped
2.  Choose the least loaded among the above selected clusters
Exception: if imbalance is greater than a given threshold, then ignore rule 1
```

Of course, this scheme requires that the steering logic monitors the availability of the operands at the time it distributes instructions. It is possible for some implementation that moving register availability information from the issue logic to the front-end cannot be done instantly, but after some delay. In that case, the steering logic will operate with partially outdated register availability information. The only effect of this is that, for a very small number of instructions, the steering will wrongly give priority to one of its source operands, but it will never affect execution correctness. We found that steering with a 1-cycle outdated availability information produces negligible performance effects for most benchmarks (0.8% IPC on average).

Figure 4-9 shows the performance of the Priority RMB compared to the Advanced RMB scheme. The Priority RMB scheme performs slightly better than the Advanced RMB for some of the benchmarks, because it reduces the latency of fetching critical operands, although the average performance improvement is small (0.2% IPC).

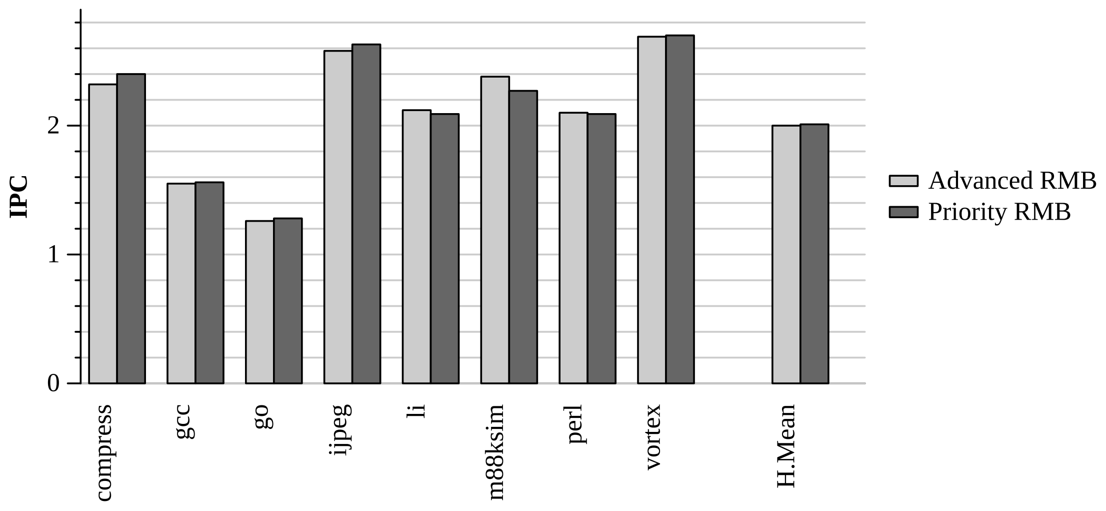

Figure 4-9: IPC of the Priority RMB vs. Advanced RMB steering schemes on a four-cluster architecture

Another approach for giving priority to critical operands could be to predict which operand will be produced last, based on previous program behavior [11]. This question opens a new research direction that has been left for future work.

4.2.7 Reducing Communications with Accurate-Rebalancing

One major drawback of the previous cluster assignment algorithm is that it generates too many communications during the periods when the workload imbalance exceeds the threshold, because then it totally ignores dependences. Moreover, the probability that the steering algorithm generates a communication in such cases grows with the number of clusters. We observed that most often, a strong imbalance situation is caused by a single overloaded cluster. Of course, rebalancing the workload does require to steer instructions to the less loaded clusters, but choosing strictly the least loaded one is probably not a critical factor. Of course, for a two-clustered organization there is no other alternative, but for four or more clusters the steering scheme could recover the strong imbalance with more accurate rebalancing actions that do not ignore completely the dependences, thus not generating as many communications.

We propose that, in case of a strong imbalance situation, instead of directly choosing the least loaded cluster, the algorithm follows the normal criteria except that the most loaded clusters are previously excluded from the choice of clusters. In doing so, there is a chance that among the non excluded clusters there is one where the source registers are mapped and thus inter-cluster communications are not required. We experimented with different exclusion criteria based on the existing signed workload counters (refer to section 4.1.2 for details), and found the most simple and effective is to exclude clusters that have a positive workload counter. Note that the Accurate Rebalancing (AR) technique could equally apply to the ARMB and the PRMB schemes proposed in this section. For the experiments in this chapter, it was assumed only the AR-PRMB scheme.

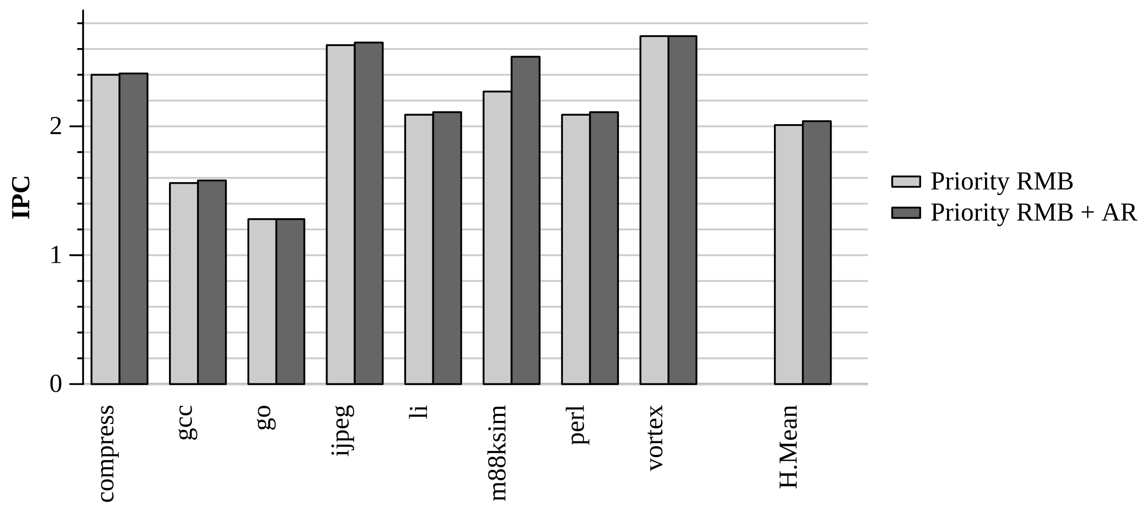

Figure 4-10: IPC of the Priority RMB steering scheme with Accurate Rebalancing, and without it, for four clusters

In more detail, this scheme works as follows:

1. To minimize communication penalties:
   1.1. If there is any unavailable operand, choose its producer cluster
   1.2. Else, select clusters with highest number of source registers mapped
2. Choose the least loaded among the above selected clusters

Exception: If imbalance > threshold, exclude clusters with workload 0, prior to applying rules 1, 2

Figure 4-10 shows the performance of the Priority RMB with Accurate Rebalancing and without it. It shows that the Accurate Rebalancing improves performance by 1.8% on average, because it reduces communications (from 0.23 to 0.20 communications per instruction, on average).

4.3 Evaluation

In this section, the RMB steering schemes are compared to the best steering schemes proposed in the literature. Next, we study the sensitivity of the various schemes to the latency and bandwidth of the interconnect. Finally, to give our conclusions a higher generality, the proposed RMB steering schemes are evaluated assuming two-cluster and four-cluster architectures, and the experiments use not only the SpecInt95 but also the Mediabench benchmark suites.

4.3.1 RMB versus Slice-Based Steering Schemes

A performance comparison of the Advanced RMB scheme against the slice-based schemes was already reported in the previous chapter (see figure 3-15), since the scheme referred to as the general balance actually matches the definition of the Advanced RMB, as already noted in section 3.3.8.

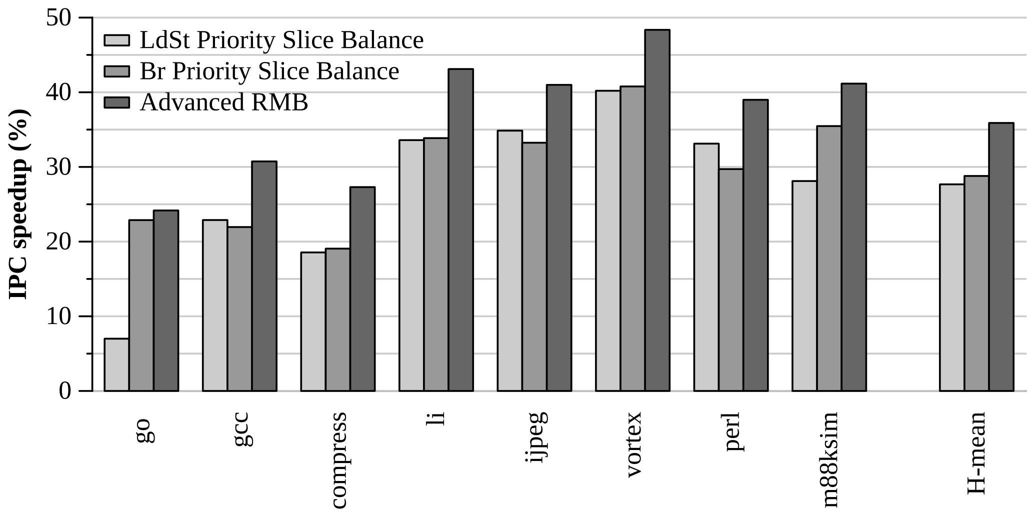

Figure 4-11: The Advanced RMB versus the best slice-based steering schemes, on a cost-effective architecture (IPC speedups over a conventional superscalar)

Figure 4-11 compares the performance of the Advanced RMB scheme (taken from figure 3-14) against that of the best slice-based scheme, i.e. the Priority Slice Balance (taken from figure 3-13), Performance is reported as the speed-up of the cost-effective two-clustered architecture over a conventional superscalar with similar complexity (see table 3-2). The results show that the Advance RMB scheme outperforms all the proposed slice-based steering schemes and achieves an average speedup of 36%. This result shows that instruction-based schemes are more efficient than slice-based ones because they work at a finer granularity, which makes them more adaptable.

4.3.2 The AR-Priority RMB versus Other Instruction-Based Schemes

We compare the Accurate Rebalancing Priority RMB scheme (AR-PRMB) to the best dynamic instruction-based approaches in the literature. It is compared to the FIFO-based scheme proposed by Palacharla, Jouppi and Smith [70,71], and also to the MOD3 scheme proposed by Baniasadi and Moshovos [10].

The basic idea of the FIFO-based scheme is to model each issue queue as if it was a collection of FIFO queues. Instructions are steered to FIFOs following an heuristic that ensures that any two consecutive instructions in a FIFO are always dependent. If all the source registers are ready, or their producers do not stay at any of the FIFO tails, then the instruction is dispatched to an empty FIFO. The assignment policy chooses always empty FIFOs from the same cluster until all of them are busy, then it switches to another cluster in round-robin order (for more details refer to the original paper [70]). In our experiments, we modelled the FIFO-based steering with 8 FIFOs per cluster, each with 4 entries (thus having twice the number of scheduler entries per cluster as our model).

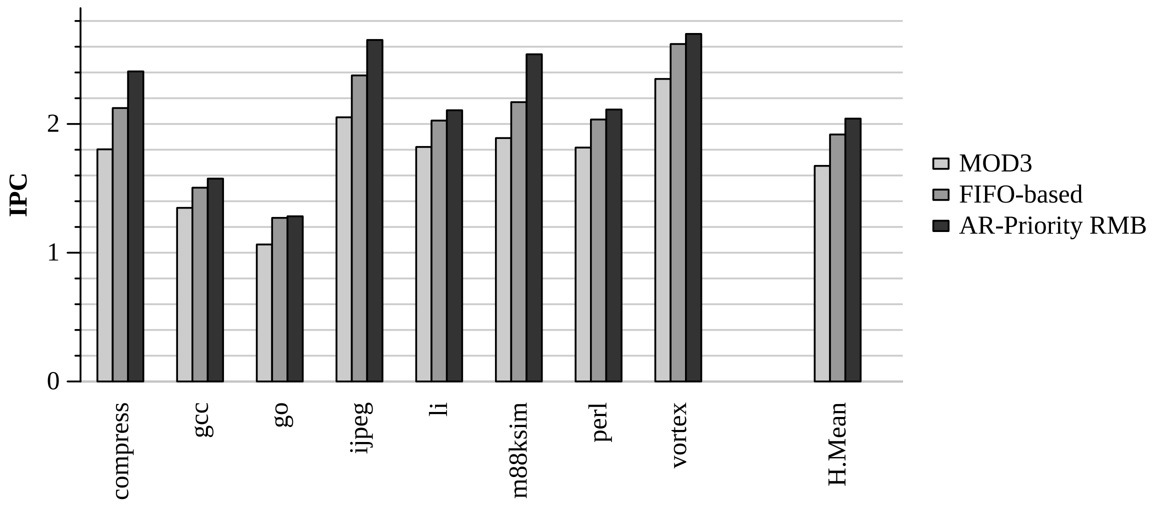

Figure 4-12: Performance of the AR-Priority RMB, FIFO-based and MOD3 steering schemes, for four clusters

Figure 4-12 shows that the AR-PRMB scheme significantly outperforms the FIFO-based steering scheme for all the programs, with an average improvement of 6.4%. The main reason for such a difference is that the FIFO-based approach generates 0.26 inter-cluster communications per dynamic instruction while the AR-PRMB generates only 0.20. Note that the FIFO-based scheme cannot drive an instruction to the cluster that produces its operands if they are ready or if the producer does not stay in a FIFO tail because another dependent instruction was previously steered.

The MOD3 cluster assignment scheme of Baniasadi and Moshovos [10] sends every group of three consecutive dynamic instructions to the same cluster, and then it switches to another cluster following a round-robin policy. Such an assignment does a very good job at keeping the workload balanced. However, it generates a much higher communication rate (0.75 communications per instruction), because it lacks a specific policy for grouping dependent instructions together. Figure 4-12 shows that our AR-PRMB scheme outperforms the MOD3 scheme by 22%.

Baniasadi and Moshovos [10] found that the MOD3 steering scheme outperforms a dependence-based scheme. Our conclusion differs from theirs, because we assume a different architecture. First, because our model assumes an additional 1-cycle overhead for issuing the copy instructions, which stresses the importance of reducing communications. Second, because they assumed a replicated register file (like the one in the Alpha 21264 [53], and the one assumed by Palacharla [70]). With such a register file, the steering logic is not concerned with dependences through ready source operands because they may have been produced well in advance, and they are probably available in all clusters. In contrast, with a distributed register file, the steering scheme must take them into account (e.g. using renaming information like in our RMB schemes) to avoid costly communication overheads.

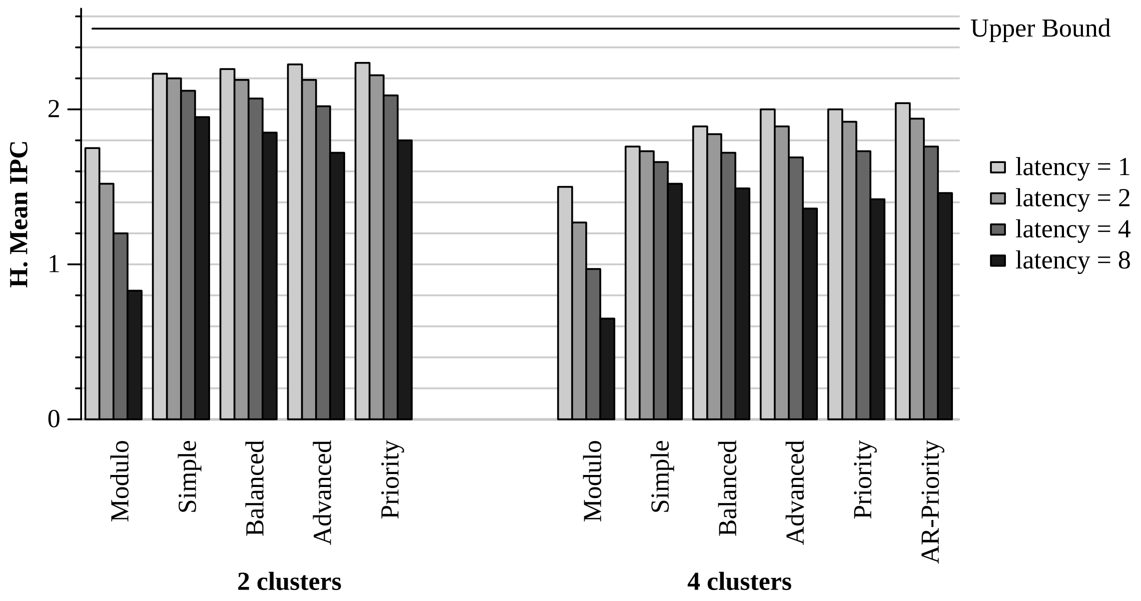

Figure 4-13: Performance sensitivity to the communication latency

4.3.3 Sensitivity to the Communication Latency

In future architectures, inter-cluster communication latency will likely increase because the interconnect wire length increases with processor complexity, and because of the widening gap between the relative speeds of gates and wires on future technologies. Using high clock rates will require not only to reduce the capacity of many components like register files and issue windows, but also to pipeline more deeply the access to other structures.

In previous sections we have assumed that inter-cluster communications take 1 cycle (there is a 1 cycle “bubble” between the copy instruction and the dependent instruction, in a different cluster). In this section, we study the sensitivity of clustered architectures to the communication latency, measured by the IPC degradation caused by a communication latency of 1, 2, 4, and 8 cycles. In all cases we assume that communications are fully pipelined, that is, for a given bypass path, one communication may begin per cycle regardless of its total latency.

Figure 4-13 shows the performance of the steering schemes presented in section 4.2 for a range of communication latencies between 1 and 8 cycles, with two and four clusters. Of course, the results for a 1 cycle latency and 4 clusters are the same as those presented in section 4.2.

As expected, the performance loss is higher with four clusters than two, due to the higher communication rate. We can also observe that, as latency increases, there is a growing performance loss and this loss is much greater for the modulo scheme than for the others, since it does not take into account instruction dependences. This behavior stresses the increasing importance of minimizing the number of communications. We can also see significant differences among the five RMB schemes: for small latencies, the AR-Priority, the Priority and the Advanced schemes outperform the other two because they achieve a better trade-off between communications and workload balance. However, as latency increases beyond 4 cycles, they perform worse because the performance loss is dominated by the communication penalty, and it is no longer worth trading communication for workload balance.

For high latencies, the performance of all the analyzed schemes is quite below the upper bound of the centralized architecture. In this scenario, the steering schemes should make more emphasis in reducing the number of communications, even at the expense of a worse workload balance. Alternatively, other communication saving techniques could be implemented, such as those proposed in the next chapter.

4.3.4 Sensitivity to the Interconnect Bandwidth

The inter-cluster communication bandwidth has a direct impact on the complexity and delay of the register files [28] and the bypass network, since it determines the number of register file write ports devoted to remote accesses, the number of bypass multiplexer’s inputs coming from remote clusters, and the number of outputs from the bypass network to the interconnection network. Furthermore, the inter-cluster communication bandwidth also determines the number of tags that are broadcast to the instruction queues of remote clusters. Therefore, it has a direct impact on the complexity and delay of the wake-up logic, which depends quadratically on the total number of tags crossing its CAM cells [70, 71].

So far, we have assumed an unbounded bandwidth for the interconnection network to isolate our results from possible communication bandwidth bottlenecks. Here we study the negative impact of contention delays when assuming a limited bandwidth. For an N-cluster configuration, we assume a simplified model with NxB independent paths. Each path is implemented through a pipelined bus where any cluster can send a value and each bus is connected to the write port of a single cluster register file. Therefore, we assume that each register file has B write ports for inter-cluster communications. Any cluster may allocate one of these paths to write a value to a remote register file, and holds it during a single cycle, since the communication is fully pipelined. Obviously, this model is somewhat idealized, since it omits the complexities due to pipelining, arbitration, or variable latencies dependent on the topology, but it may provide a first order approach to evaluate the problem.

We evaluated the performance sensitivity to the interconnect bandwidth, within a range of 1 to 4 incoming message per cluster each cycle and also with an unlimited bandwidth on a four-clustered architecture. The results in Figure 4-14 correspond to the AR-Priority RMB steering scheme, and they show that the performance degradation caused by contention delays when the communication bandwidth is reduced to a single incoming path per cluster is almost negligible. Although not shown, similar results were observed with the rest of RMB schemes. This result is consistent with the low number of communications per instruction of the RMB schemes, as presented in previous sections (and also summarized in figure 4-16 below). We also found that only the modulo steering achieves a significant performance improvement as bandwidth increases, because of its huge communication demands (see figure 4-2).

In consequence, these results suggest that for inter-cluster communications on a clustered architecture with an RMB steering scheme it may suffice just a single write port in each register

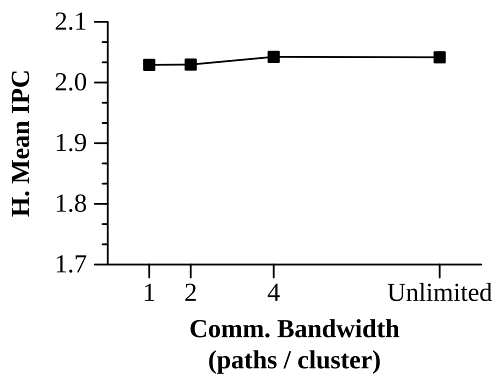

Figure 4-14: Sensitivity to the interconnect bandwidth for four clusters (AR-Priority RMB steering)

file, a single incoming tag per issue window, and a single remote bypass attached to the input multiplexers of the functional units. Moreover, the interconnect can be implemented with low hardware cost and complexity. Several proposals and a more in-depth study of the interconnect network will be conducted in Chapter 6.

4.3.5 Overall Evaluation of the RMB Steering Schemes

In section 4.2, performance results were reported individually for each RMB steering scheme Here, all of them are gathered in a single graph to allow a direct comparison, and the scope of the experimental assumptions is extended, for the sake of a higher generality. Two eight-way issue superscalar architectures are studied, with two and four clusters respectively (see parameters in table 2-1). In addition, the experiments shown below use both the SpecInt95 and the Mediabench benchmark suites (see details in section 2.3).

Figure 4-15 compares the harmonic mean IPCs for the six steering schemes. Figure 4-16 compares the average number of communications per instruction, and figure 4-17 compares the average NREADY workload imbalance metric. In all cases, the best performing heuristics is the AR-Priority RMB scheme. The results with two clusters show similar trends as those with four clusters, although the differences among the various steering schemes are smaller, because of the lower communications and workload imbalance overheads (figures 4-16 and 4-17).

The results with the Mediabench benchmark suite show similar trends as those studied with the SpecInt95, but the differences among the various steering schemes are more prominent (the AR-Priority RMB scheme for four clusters outperforms the Simple RMB scheme by 32%, while it was only 16% with the SpecInt95). The Mediabench programs have more ILP than the SpecInt95, which make them more prone to suffer high workload imbalances. Therefore, improving the workload balancing ability of the steering also has a greater positive impact on performance.

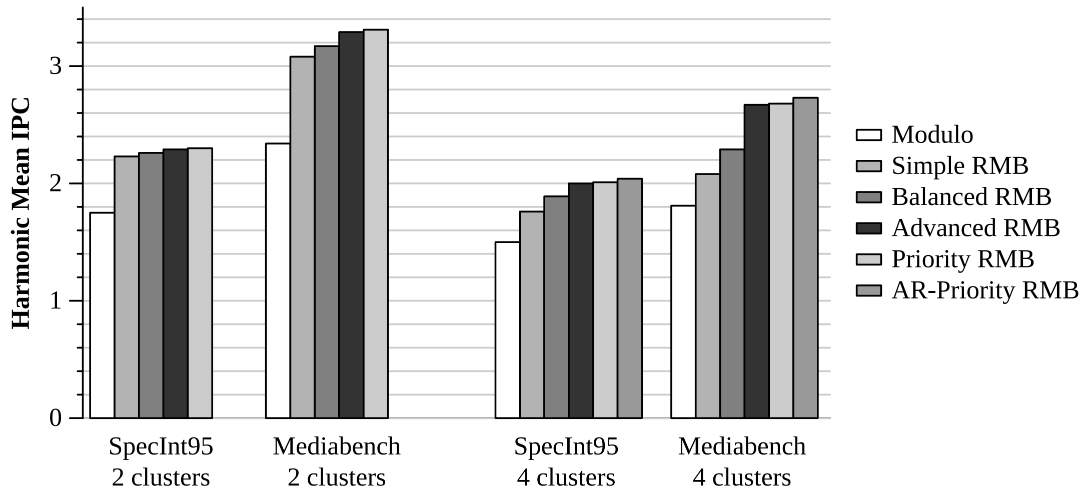

Figure 4-15: Overall performance of the RMB steering schemes

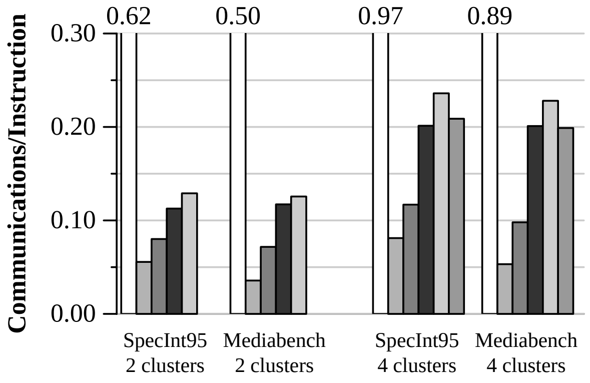

Figure 4-16: Average number of communications per instruction

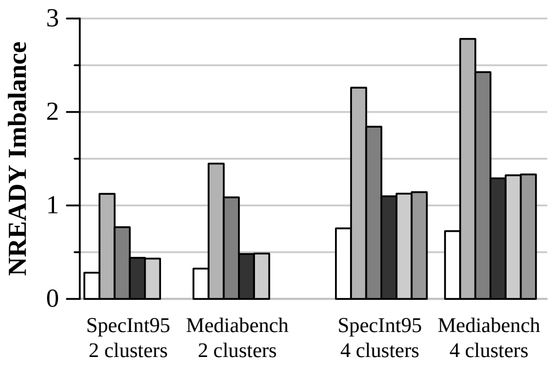

Figure 4-17: Average NREADY workload imbalance

4.4 Conclusions

We have proposed four new instruction-based mechanisms that dynamically partition a sequential program into the different clusters of a clustered microarchitecture. Instruction-based schemes perform finer-grain cluster assignments than slice-based ones, on a per-instruction basis, so they are more flexible and more effective than slice-based schemes. The proposed RMB schemes follow a primary and secondary criteria that specifically address the goals of minimizing communications and maximizing workload balance respectively.

We have proposed a new workload balance metric (NREADY) that matches the intuitive concept of workload balance and, unlike previous proposals, it is suitable for any number of clusters. Likewise, we propose a new imbalance counter mechanism, suitable for any number of clusters (DCOUNT), which is used to guide efficiently the steering decisions of the proposed RMB schemes.

Our experiments show that the proposed AR-Priority RMB steering scheme is more effective than all the existing slice-based schemes, either static or dynamic, and it significantly outperforms the best dynamic schemes previously proposed in the literature, because it achieves the best trade-off between communications and workload balance.

Finally, we found that a clustered architecture is quite sensitive to the inter-cluster communication latency, which emphasizes the importance of reducing communications with appropriate steering decisions. However, we found that, with our RMB steering schemes, it is hardly sensitive to the cluster interconnect bandwidth, because it has a low bandwidth demand. Therefore, inter-cluster communications can be implemented with very low complexity impact on the register files, the issue windows, the bypasses, and the cluster interconnect networks, which results in short cycle times and low power consumption.