**Multicluster Architecture**

- Hardware is partitioned.
- Each partition (cluster) has its own register files, dispatch queues, and functional units.

**Motivation**
To reduce clock cycle time by reducing the size and complexity of the components on critical timing paths.

**Advantages**
- Each cluster issues fewer instructions per cycle.
- Register files require less read/write ports.
- Less complex instruction-scheduling logic.

**However**
- Resource partitioning has two major drawbacks.
  1) Instruction execution overhead is increased.
  2) Concurrently executing instructions may be reduced.

- To counter these negative drawbacks, Static Instruction Scheduling algorithm introduced.

**Challenge: Complexity vs speed**
- Balancing the complexity of hardware and the speed at which this hardware can be clocked.
- This challenge exists because:
  
  - First: complexity of a component increases its cycle time and thus that of the processor.
  - Second: complex components can be larger and further apart, thus the time for signals to travel between them might be greater, thereby increasing the processors cycle time.

**Architecture**

*Figure 1: A dual-cluster processor built within the multicluster architecture framework.*
Separate Dispatch Queues and Register Renaming Hw

- This architecture implements dynamic scheduling using dispatch queues and register renaming hardware.
- Dispatch queue maintains pool of instructions coming from instruction scheduler.
- Register renaming hardware maps architectural registers to a larger set of physical registers.
- The instructions are obtained from single instruction cache. Also data cache is shared by all clusters.

Instruction Distribution and Execution

- Instructions are fetched and distributed in order.
- If an instruction can’t be distributed to a cluster because a dispatch-queue entry or a physical register is not available, instruction queue stalled until required source become available.
- Local register is the architectural register assigned to a specific cluster. Only one register is assigned.
- Global register is the register assigned to both clusters. Thus two registers are assigned.
- Global registers are generally used for stack and global pointers.

Scenario 1

- All operands mapped to same cluster. So that cluster is issued.
- Output register is renamed.
- Operands are mapped.
- Instruction is executed and result is written back as usual.

Scenario 2

![Diagram showing execution of instruction r2 ← r0 + r1 when operands r0 is forwarded to cluster C1.]

Scenario 3

![Diagram showing execution of instruction r2 ← r0 + r1 when result is forwarded to cluster C2.]

Scenario 4

![Diagram showing execution of instruction r2 ← r0 + r1 when global result is forwarded to cluster C2.]

Figure 2: Dual execution of the instruction r2 ← r0 + r1 when operands r0 is forwarded to cluster C1.

Figure 3: Dual execution of the instruction r2 ← r0 + r1 when result is forwarded to cluster C2.

Figure 4: Dual execution of the instruction r2 ← r0 + r1 when global result is forwarded to cluster C2.
**Scenario 5**

Inter-cluster communication and Workload

- There are two main issues concerning cluster architectures;
- Since inter-cluster communications can easily take one or more cycles delay introduced to dependent functions (throughput decreased, CPI is also increased).
- Second issue is the workload balance.
- In order to achieve highest performance we have to balance workload optimally at the same time minimize the number of communications.
- Programs can be partitioned either in compile time (statically) or at run-time (dynamically).

**Static Instruction Scheduling**

- 1) Code optimization: Application is compiled into an intermediate Language (IL).
- 2) Code Scheduling: The IL instructions are arranged into static code scheduling. (Trace scheduling is used)
- 3) Candidates for local and global registers are determined.
- 4) Live range partitioning: Live ranges partitioned to maximize concurrent utilization and minimize dual-distributed instructions.
- 5) Register allocation: Live ranges allocated to architectural registers. (Graph coloring technique is used).
- 6) Machine level instructions are arranged into a code schedule.

**Live Range Partitioning**

- To determine cluster assignments for the live ranges;
- Local scheduler sorts the basic blocks according to estimates.
- Basic blocks with equal estimates are sorted by the number of static instructions.
- Bottom-up inorder traversal is carried out.
- Block order: 4 1 5 3 2
- Live range order: C G B A E D H

<table>
<thead>
<tr>
<th>benchmark</th>
<th>speedup ratio (1)</th>
<th>speedup ratio (2)</th>
<th>speedup ratio (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>compress</td>
<td>-14</td>
<td>-6</td>
<td></td>
</tr>
<tr>
<td>dodge</td>
<td>-21</td>
<td>-15</td>
<td></td>
</tr>
<tr>
<td>gcc1</td>
<td>-15</td>
<td>-16</td>
<td></td>
</tr>
<tr>
<td>gun</td>
<td>-5</td>
<td>-22</td>
<td></td>
</tr>
<tr>
<td>sub100</td>
<td>-36</td>
<td>-25</td>
<td></td>
</tr>
<tr>
<td>toc</td>
<td>-41</td>
<td>-19</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. The speedup ratios 100\((1 - \frac{1}{N_{ranges}})\) obtained when the benchmarks were not rescheduled (column 2) and when the local scheduler was used to rescheduling them (column 3).
Results

- Local scheduler significantly reduces the slow down.
- Furthermore in the case of compress performs better than single-cluster.
- This is due to larger dispatch queue with branch miss-prediction.
- Worst case slow down is $\%25$.
- To compensate for the increase in clock cycles dual processor would have to use processor clock with period $\%20$ slower.
- $20 = 100 - 100XN_{single}/1.25XN_{single}$

Delays & Conclusion I

- $0.35 \text{ micrometer } \% 18$
- $0.18 \text{ micrometer } \% 82$
- Given that there is only an $18\%$ difference between the cycle times of the $4$-issue and $8$-issue processors in a $0.35 \text{ micrometer process generation clustering}$ would not help much.
- However for a $0.18 \text{ micrometer generation worst}$ case path would increase by $82\%$. The larger delays is due to communication delays rather than computation delays.
- So Clustering could result in a significant overall performance.

Cost Effective Clustered Architecture

- Idea
- In current superscalar processors, all FP resources are idle during the execution of integer programs.
- This problem can be alleviated if the FP cluster is extended to execute simple integer instructions. The issue width can potentially be doubled without increasing the complexity.
- In fact the result is a clustered architecture with two heterogeneous clusters.

Abstract

- Dynamic Steering Logic is used.
- The evaluation showed an average speedup of $35\%$ over a conventional $8$-way issue ($4 \text{ int} + 4 \text{ fp}$) machine.
- Although integer programs seem not to have much parallelism, there is a growing number of integer applications with high ILP such as multimedia workloads (real time video and audio).

Static vs Dynamic partitioning

- A static partitioning means that the compiler tags each instruction according to the cluster in which it will be executed. Main advantage is minimal hardware support is required.
- Drawbacks of this scheme is the ISA of the processor has to be extended.
- In dynamic scheme ISA has not to be changed and therefore the clustering of a processor is transparent to applications running on it.
- The dynamic scheme is also more adaptable to the state of the pipeline since it can decide where an instruction will be executed according to the actual state of the pipeline.
- Steering logic send dependent instructions to the same cluster.
Dynamic Steering Logic

- An estimation of workload could be the number of instructions in its instruction queue.
- However this measure does not take into account the parallelism among these instructions. For instance if one cluster has many instructions but each depends on the previous one it would be quite idle.
- Thus a better estimation of the workload is the number of ready instructions which also reflects the parallelism.
- Balance is the second factor and computed every cycle and the average of last N cycles is used by steering logic.

Steering Schemes

- RMBS Simple Register Mapping Based Steering
  - If instruction has no source operands it is randomly assigned to a cluster.
  - If instruction has one source operand it is assigned to the cluster in which operand is mapped.
  - If the instruction has two source operands and both are in the same cluster it is sent to same cluster otherwise randomly sent.

Balanced RMBS

- If no source operands; assigned to least loaded cluster.
- If the instruction has one operand it is assigned to cluster in which the operand is mapped.
- If the instruction has two operands and both are in the same cluster it is sent to same cluster otherwise its assigned to least loaded cluster.

Advanced RMBS & Modulo Steering

- If there is significant imbalance the instruction is assigned to least balanced cluster.
- Otherwise it does the same as the Balanced scheme.
- Modulo Steering
  - This scheme consists of sending alternatively one instruction to each cluster if the instruction can be executed in both clusters. If it can just be executed in one of the clusters it is steered to that cluster.

### Parameter Configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch width</td>
<td>6 instructions</td>
</tr>
<tr>
<td>Dec/Issue width</td>
<td>2 instructions</td>
</tr>
<tr>
<td>Instruction queue size</td>
<td>64</td>
</tr>
<tr>
<td>Min. to high instructions</td>
<td></td>
</tr>
<tr>
<td>Max. to low instructions</td>
<td></td>
</tr>
<tr>
<td>Issue width</td>
<td>8 instructions</td>
</tr>
<tr>
<td>Functional units</td>
<td>3 ALU + 1 mul, 1 ALU + 1 mem, 1 branch unit</td>
</tr>
<tr>
<td>Instruction queue size</td>
<td>32</td>
</tr>
<tr>
<td>Memory space</td>
<td>4 instructions</td>
</tr>
<tr>
<td>Loads and store on 1st</td>
<td>4 instructions</td>
</tr>
<tr>
<td>64K cache</td>
<td>56</td>
</tr>
</tbody>
</table>

### Table 3: Machine Parameters (separate cluster 1 and cluster 2 if not common)

![Modulo 5% Simple RMBS %22 Balanced RMBS %27 UB %44 Upper Bound](image-url)
As expected Modulo has largest communication penalties and explains why it is the slowest.

Least balanced is the Simple RMBS since it does not implement any balance policy. Balanced RMBS is better. Modulo is the best and nearly optimal.

If there was 0 comm latency

Introducing Advanced RMBS to improve workload a 35 improvement is achieved.

In future communication latencies will likely be longer than one cycle. Therefore the impact of communication latency is analysed.
Applicability to FP programs.

Dynamic versus static steering.

Related Work

- Clustering is also present in some multithreading architectures. The criteria used to partition programs is based on control-dependences instead of data dependences.
- Clustering can also be applied to VLIW architectures but they perform cluster assignment at compile time.

Multimedia

- Although multimedia programs are not targeted in this article; the performance improvement achieved with this architecture is significant.

Conclusion II

- Evaluation shows that it can be achieved %35 speedups on the average over a conventional 8 way issue (4 int + 4fp) machine.
- Hence with minimal hw support and no ISA change idle fp resources can be profitably exploited to speed-up integer programs.