氏名（本籍地）	高屋敷 光（岩手県）
学位の種類	博　士（情報科学）
学位記番号	情　博　第　７９４　号
学位授与年月日	令和　５年　３月２４日
学位授与の要件	学位規則第４条第１項該当
研究科、専攻	東北大学大学院情報科学研究科（博士課程）情報基礎科学専攻
学位論文題目	Latency-tolerant Vector Processor Architectures
論文審査委員	（主査）東北大学教　授　小林 広明
	東北大学教　授　青木 孝文
	東北大学准教授　佐藤 雅之
	東北大学准教授　小松 一彦
	(サイバーサイエンスセンター)

論 文 内 容 の 要 旨

Chapter 1 Introduction

A vector instruction set is now available for modern processors. This instruction set contributes to gaining high computing capability. As a result, the vector instruction set is essential for modern processors.

In combination with the high memory performance, vector processors that utilize a vector instruction set can achieve high sustained performance. By optimizing the architecture for vector instructions with a long vector length and designing the memory system to prioritize high memory bandwidth, vector processors can achieve a high performance in scientific and engineering applications. The increasing demand for higher accuracy and broader applicability in these fields has led to a growth in performance requirements, making vector processors a promising option for achieving high performance.

This dissertation aims to establish the architectonic methodology for vector processors that enables high sustained performance in memory-intensive applications from the viewpoints of latency-tolerance. The latency tolerance to be discussed in this dissertation is defined as the ability to achieve a high sustained performance in the case of irregular memory access or indirect memory access in memory-intensive applications. The latency-tolerant architectures can contribute to the resolution of latency-related problems when designing the next-generation vector processors.

Chapter 2 Importance of latency-tolerance for vector processor architectures

Modern vector processors have adopted four features different from the classical ones. First, modern vector processors have increased their computing capability by increasing the number of vector cores, even though historically the vector processors have focused on increasing the performance of one vector core. As this trend may continue in the future, the many-core technology will play an essential role in driving the growth of computing capability of vector processors. Second, modern vector processors employ an out-of-order execution mechanism for vector instructions. This allows vector processors to increase the utilization of vector functional units, as well as the ability to exploit instruction-level parallelism for higher sustained performance. Third, modern vector processors have a virtual memory system. This system separates the memory space perceived by processes from the actual memory space, leading to improved memory usage efficiency and reduced programming workload for the user. Fourth, modern vector processors utilize multi-banked caches in order to maintain high sustained memory bandwidth. As the number of input/output pins that can be integrated onto a single chip is limited, further improving the off-chip memory bandwidth becomes difficult. Therefore, the cache is used to provide vector cores with reusable data at high bandwidth

rather than relying on the off-chip memory.

Applications for the modern vector processors are actively developed to improve accuracy and expanded their scope. Recently, new applications such as graph processing and machine learning, which involve irregular memory accesses, have become popular workloads. Since the modern vector processors and their memory systems are optimized for high sustained performance in the case of continuous memory accesses, the processors may not perform at their best in the case of irregular ones. Furthermore, irregular memory accesses may cause cache misses due to the lack of locality for the data reference, making it difficult for vector processors to maintain high performance in applications with irregular memory accesses. One common aspect of these problems is latency, i.e., the time between issuing a vector instruction and its completion. The latency is a common issue that affects the performance of systems dealing with irregular memory accesses. It can arise due to delays in accessing the data or the overhead of managing the accesses. Reducing the latency is often a key factor in improving the performance of systems with irregular memory accesses.

It is generally said that the latency problem does not arise in vector processors because their vector processing mechanism can hide latency by pipelined data accesses. However, in modern vector processors, the latency problem becomes a performance bottleneck due to the following four reasons. First, as semiconductor manufacturing technologies improve computing performance, the time required for each vector operation tends to become short. The latency hiding capability of vector processors is gradually decreasing. Second, the number of instructions that the processor can handle simultaneously may be insufficient to exploit instruction-level parallelism from the program. For example, irregular memory accesses can be handled by the vector instructions set using specialized instructions. This handling requires multiple instructions for one access, resulting in a longer latency than sequential accesses. Third, in applications with irregular memory accesses, data are not reused in the cache system properly. The irregular memory accesses may not have locality in data references, the cache system cannot fully be utilized. To solve the problems related to these reasons, this dissertation proposes four approaches.

Chapter 3 Indirect memory access prefetcher for vector gather instructions

An indirect memory access prefetcher for vector gather instructions is designed to reduce the latency of indirect memory accesses on modern vector processors. These accesses are implemented as vector gather instructions in the vector instruction set, which allows for the vectorization of irregular or sparse memory access sequences. When a vector gather instruction is executed, it loads data from the main memory based on index data that has been previously loaded. Because the values in the index data are unpredictable, the processor must wait for their arrivals, which can result in long memory access latencies that negatively impact performance even on vector processors. To solve this issue, the vector gather prefetcher employs a two-phase approach. In the first phase, the vector gather prefetcher loads the index data with assuming that the index data will be accessed sequentially. In the second phase, after prefetching the index data is complete, the vector gather prefetcher predicts the addresses for the indirect memory accesses using the prefetched index data and attempts to prefetch the data at those addresses. This allows the prefetcher to hide the latency caused by indirect memory accesses.

The evaluation of the vector gather prefetcher is performed by using a simple kernel with two types of index data: sequential values and random values. The evaluation results show that the prefetching mechanism improves the performance of the sequential-indexed and random-indexed kernels by 2.2× and 1.2×, respectively.

Chapter 4 Criticality-aware out-of-order mechanism for vector gather instructions

A criticality-aware out-of-order mechanism aims at hiding latency of instructions that cause stalling. The conventional out-of-order mechanism has a limitation in the number of instructions that can overtake other instructions. If many instructions with dependencies run out the out-of-order window, the processor stops issuing instructions. Thus, the criticality-aware out-of-order mechanism issues the instructions causing stalling on another

execution path. This mechanism enables modern vector processors to exploit more instruction-level parallelism and conceal the latency of issuing instruction.

The evaluation of the criticality-aware out-of-order mechanism is performed by using memory-intensive applications such as numerical simulations and graph applications. The evaluation results show that the proposed mechanism achieves up to 80% performance improvements on several memory-intensive applications.

Chapter 5 Page-address coalescing method of vector gather instructions

A page-address coalescing method aims at reducing the translation time between virtual memory addresses and physical memory addresses for vector instructions. Since a vector instruction handles multiple elements in one instruction, the processor has to translate multiple addresses. Furthermore, in the case of vector gather instructions that are responsible to indirect memory accesses, the processor has to translate all the addresses. This causes a long latency for address translations. The page-address coalescing method tries to deduplicate page addresses in a vector using vector arithmetic units already built in the processor.

In the evaluation of the page-address coalescing method, simulation experiments are conducted to evaluate the performance improvement on the numerical applications and the graph applications that contain many vector gather instructions. The evaluation results show that the proposed method can achieve a 2x performance improvement in numerical applications and 1.08x in graph applications.

Chapter 6 Skewed multi-banked cache for many-core vector processors

A skewed multi-banked cache is designed to reduce conflict misses that occur when multiple vector cores access the cache. While a cache with a high associativity can avoid conflict misses, implementing such a cache for a multi-banked configuration can be cost-prohibitive. Instead of increasing the associativity, the skewed multi-banked cache reduces the number of conflict misses by preventing data requests from using the same cache set. This dissertation also examines the use of several hashing functions and replacement policies in the skewed multi-banked cache. Eventually, the proposed cache employs odd-multiplier displacement hashing to effectively skew the data, and the static re-reference interval prediction policy for efficient replacement. This mechanism enables processors to achieve a high cache hit ratio and reduce latency.

In the evaluation of the skewed multi-banked cache, this dissertation discusses the cache hit rates by using stencil calculation kernels varying the number of vector cores sharing a cache and its associativity. In a 3D 7-point stencil kernel, the skewed multi-banked cache can approximately eliminate 70 % of the conflict misses. In a 3D 13-point stencil kernel, the skewed multi-banked cache can approximately eliminate 90 % of the conflict misses.

Chapter 7 Conclusions

In conclusion, this dissertation demonstrates that the proposed four approaches can solve the latency problems for modern vector processors. These approaches allow modern vector processors to realize high sustained performance in memory-intensive applications by making vector processors more tolerant to latency. These approaches to solve latency problems are valuable for architects to design vector processors in the future.

As future work, it may be worthwhile to evaluate all the proposed methods combined together in a single vector architecture. It would also be interesting to assess the effectiveness of the proposed methods applied to vector scatter instructions that perform writes using indirect memory accesses and are often used vector gather instructions together, while this dissertation primarily focused on vector gather instructions with regard to indirect memory accesses.

論文審査結果の要旨

ベクトルプロセッサは単一命令で複数のデータを扱うベクトル命令に特化したプロセッサであり，高バンド幅メモリシステムと組み合わせることにより、アプリケーションの実行において高い実効性能を発揮する．しかしながら，疎行列処理やグラフ問題などデータ依存性が高く不規則なメモリアクセスを要するアプリケーションの広がりに伴い，命令処理やデータアクセスの長レイテンシがベクトル処理の効率を妨げる要因となりつつある．本論文は，この問題を解決するレイテンシ耐性を備えるベクトルプロセッサアーキテクチャについて論じたものであり，全編 7 章からなる．

第 1 章は緒論である．

第 2 章では，ベクトルプロセッサにおいてレイテンシ耐性の重要性を明らかにしている．ベクトルプロセッサを取り巻く状況について精査し，レイテンシの増大がベクトル処理の実効性能のボトルネックになることを示している．さらにレイテンシを増加させる要因を明らかにし，レイテンシ耐性を実現するための 4 つの指針を明らかにしている．これらは，レイテンシ耐性を有するベクトルプロセッサの設計指針を与える重要な知見である．

第 3 章では，1 つ目の指針に基づきベクトル間接参照のためのプリフェッチ機構を提案している．本機構は,ベクトル間接参照を構成する 2 種類のベクトルメモリ命令の実行において，そのメモリアドレスを予測してキャッシュに予めデータを保存する．本プリフェッチ機構をベクトルプロセッサに適用するためにパラメータの最適化を行い，その有効性を実験により明らかにしている．

第 4 章では，2 つ目の指針に基づき，ベクトル命令発行レイテンシを削減する機構を提案している．本提案はベクトル命令発行レイテンシを削減するためにベクトル命令の投機的なアウトオブオーダー実行を行う. それと同時に, 投機的に得られた命令実行結果を再利用することにより，実効メモリバンド幅の向上も可能とする．本提案は高いメモリバンド幅を活用するベクトルプロセッサとの親和性が非常に高く，有益な成果である．

第 5 章では，3 つ目の指針に基づきベクトル間接参照における冗長なアドレス変換を排除する手法を提案している．本提案によりベクトル間接参照命令のアドレス変換にかかるレイテンシを削減可能である．また，冗長排除には既存のベクトル演算ユニットを活用してハードウェアコストを抑制している．これらのことは，ハードウェアの有効活用と実効性能向上に関して優れた成果である．

第 6 章では，4 つ目の指針に基づき，スキュー型高バンド幅多バンクキャッシュメモリを提案している．本提案により比較的低い連想度を有するセットアソシアティブキャッシュにおいて顕著となるコンフリクトミスを削減し，その実効性能の向上への貢献を実験により明らかにしている．これらは重要な成果である．

第 7 章は，本論文を総括し，結論としている．

以上要するに本論文は，複雑なデータ依存関係を有し不規則なメモリアクセスを伴うアプリケーションに対しても、高いベクトル処理能力を実現するための重要な知見を与えたもので，情報基礎科学および計算機科学の発展に寄与するところが少なくない．

よって，本論文は博士（情報科学）の学位論文として合格と認める．