研究課題別評価

1 研究課題名：スケルトン並列プログラムの最適化

2 研究者氏名：胡 振江 (HU, Zhenjiang)

3 研究の狙い：

With the increasing popularity of parallel programming environment such as PC cluster, more and more people, including those who have little knowledge about parallel architectures and parallel programming, are hoping to write parallel programs to solve their daily problems. This situation eagerly calls for models and methodologies that can assist programming parallel computers effectively and correctly.

Data parallel model turns out to be one of the most successful ones for programming massively parallel computers. To support parallel programming, this model basically consists of two parts: (1) a parallel data structure to model a uniform collection of data which can be organized in a way that each element can be manipulated in parallel; and (2) a fixed set of parallel skeletons on the parallel data structure to abstract parallel computation structures of interest, which can be used as building blocks to write parallel programs. Typically, these skeletons include element-wise arithmetic and logic operations, reductions, prescans, and data broadcasting.

This data parallel model not only provides programmers an easily understandable view of a single execution stream of a parallel program, but also makes the parallelizing process easier because of explicit parallelism of the skeletons. For instance, in high performance Fortran 90/95, the parallel data structure is array and the parallel skeleton is FORALL; in the parallel language NESL, the parallel data structure is sequence and the most important parallel skeletons on sequences are apply-to-each and scan; and in the BMF (Bird Meertens Formalisms) parallel model, the parallel data structure is typically parallel list, and the parallel skeletons are mainly map and reduce.

Despite these promising features, the application of current data parallel programming using skeletons suffers from several problems, which prevent it from being practically used. Firstly, because parallel programming relies on a set of parallel primitive skeletons for specifying parallelism, programmers often find it hard to choose proper ones and to integrate them well in order to develop efficient parallel programs to solve their problems. Secondly, the skeletal parallel programs are difficult to be optimized, and the major difficulty lies in the construction of rules meeting the skeleton-closed requirement for transformation among skeletons. Thirdly, skeletons are assumed to manipulate regular data structures. For irregular data structures like nested lists where the sizes of inner lists are much different, the parallel semantics of skeletons would lead to load unbalance which may cancel the effect of parallelism in skeletons.

This project aims to solve these problems based on the theory of Constructive Algorithmics, investigating what kinds of recursive structures are suitable for capturing parallel computation on parallel data structures, and constructing rules for manipulating such recursive structures.

4 研究成果：

Our main contribution is a novel framework supporting efficient parallel programming using skeletons. We have designed and implemented a self-optimizing C++ parallel skeleton library, with which users, with little knowledge of parallel programming, can program a wide class of parallel algorithms without the need to be concerned with details of parallel architectures and data communications among processors. In addition, we have proposed a programming methodology for systematic development of efficient skeletal parallel programs. In the following, we shall detail three important results.

4．1 A Generic Parallel Skeleton for Parallel Programming

We have proposed a new powerful parallel skeleton, which can significantly ease skeletal parallel programming, efficiently manipulate both regular and irregular data, and systematically optimize skeletal parallel programs.

We have defined a novel parallel skeleton that cannot only efficiently describe data dependency in a computation through an accumulating parameter, but also exhibit nice algebraic properties for manipulation. It can be considered as a higher order list homomorphism, which abstracts a computation requiring more than one pass and provides a better recursive interface for parallel programming.

We have given a single but general fusion rule, based on which we construct a framework for systematically optimizing skeletal parallel programs. Inspired by the success of the shortcut deforestation for optimizing sequential functional programs in compilers, we gave a specific shortcut law for fusing composition of skeletal parallel programs, but paying much more attention to guaranteeing the skeleton-closed property. Our approach using a single rule is in sharp contrast to the existing approaches based on a huge set of transformation rules developed in a rather ad-hoc way. Furthermore, we proposed a flattening rule to deal with both regular and irregular nested data structures efficiently. Compared to the work by Blelloch where the so-called segmented scan is proposed to deal with irregular data, our rule is more general and powerful, and can be used to systematically handle a wider class of skeletal parallel programs.

4．2 A Self-Optimizing Skeleton Library

We have designed and implemented a parallel skeleton library that guarantees efficient combinations of skeletons. Our idea was to associate each skeleton not only with an efficient parallel implementation but also with an interface for efficient combination with other skeletons. This interface contains information about how the skeleton consumes and produces its data. This idea is not new in the functional community, where we have seen the success of shortcut deforestation (fusion) in optimizing sequential programs in compilers. However, as far as we know, we are the first to introduce this idea to the design of parallel skeleton libraries.

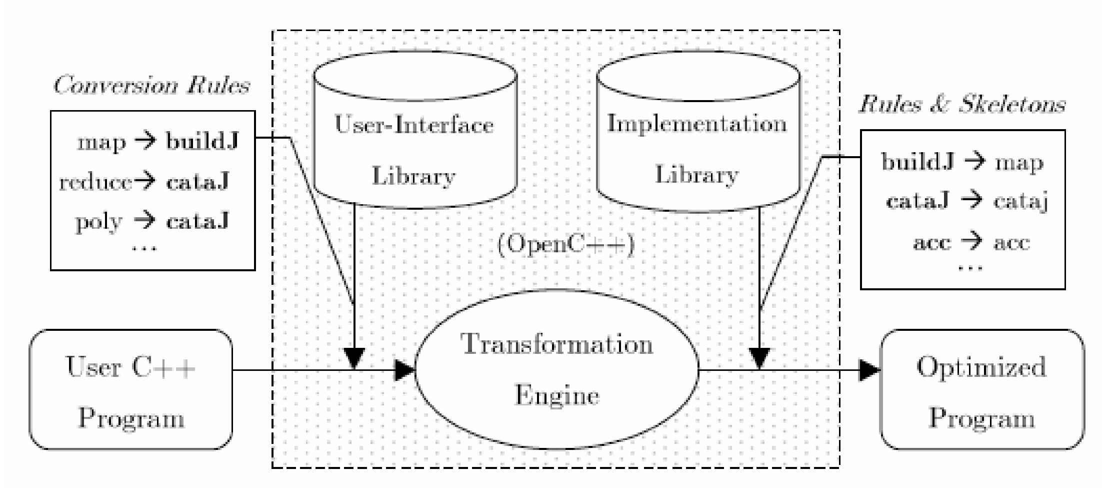

Figure 1: A Fusion-Embedded Skeleton Library

Our skeleton library for skeletal parallel programming in C++ (Figure 1) has the following new features. First, we need only a single optimization rule, and this rule can be applied to skeletal parallel programs in any way while guaranteeing the same result and termination. Second, our library allows new skeletons to be introduced without any change to the existing optimization framework, and ensures their efficient combination with existing skeletons in the library. This remedies the situation where transformation rules must take combinations of the skeletons with existing ones into account. Third, our library is simple to use. From the programmers' point of view, as our library does not introduce any new syntax, a programmer who knows C++ should have no trouble in using it. We construct a structured interface for the skeletons as well as apply a general optimization rule concisely and quickly with the help of the reflection mechanism provided with Open C++. We found it very useful to use meta programming in implementing the transformation, which, we believe, is worth greater recognition in the skeleton community.

4．3 From List Skeletons to Tree Skeletons

Trees are useful data types, widely used for representing hierarchical structures such as mathematical expressions or structured documents like XML. Due to irregularity of tree structures, developing efficient parallel programs on trees is much more difficult than developing efficient parallel programs on lists. Unlike linear structure of lists, trees do not have a linear structure, and hence the recursive functions over trees are not linear either (in the sense that there are more than one recursive calls in the definition body). It is this nonlinearity that makes the parallel programming on trees complex.

We have given a systematic method for parallel programming using tree skeletons, by proposing two important transformations, the tree diffusion transformation and the tree context preservation transformation. The tree diffusion transformation is an extension of the list version. It shows how to decompose natural recursive programs into equivalent parallel ones in terms of tree skeletons. The tree context preservation transformation is an extension of the list version too. It shows how to derive associative operators that are required when using tree skeletons.

In addition, to show the usefulness of these theorems, we have demonstrated the first formal derivation of an efficient parallel program for solving the party planning problem using tree skeletons.

5　自己評価：

The three and half years are short, but fruitful and enjoyable. I am pleased that the results we have achieved are really encouraging; our new parallel skeleton library, together with a set of nontrivial applications, has proved that the theory of constructive algorithmics, i.e., the program calculation theory, is practically useful for better solving problems in parallel programming, which has not been well recognized so far.

Our main contribution is a novel framework supporting efficient parallel programming using skeletons. First, we have designed and implemented a C++ parallel skeleton library, with which users can code their parallel algorithms as if they used other library functions without need to concern about details of parallel architectures and data communications among processors. Second, we have proposed a programming methodology, which is useful for systematically developing efficient skeletal parallel programs from initial straightforward specifications. Third, we have implemented a programming environment, with which one can develop and run skeletal parallel programs efficiently. This framework can greatly help easing parallel programming using skeletons, efficiently manipulating both regular and irregular data, and systematically optimizing skeletal parallel programs.

It is our hope that the results of this work would lead to a future standard framework for skeletal parallel programming that can greatly help easing parallel programming using skeletons, efficiently manipulating both regular and irregular data, and systematically optimizing skeletal parallel programs. In practice, we would expect the first performance-guaranteed parallel skeleton library in C++, which is really useful for solving practical problems. We wish it to be a convincing witness of usefulness of constructive approach in parallel programming. In theory, we would expect a unified algebraic （constructive）model for structuring data, control, and communication skeletons in development of efficient parallel programs.

6　研究総括の見解：

胡氏の研究は，データに内在する並列性を並列データ構造として捉え，それを処理するための並列処理スケルトンを少数用意し，それらの組み合わせによって並列処理プログラムを作成しようとするものである．このような方式で並列プログラムを構成するための理論体系を整備し，それにもとづいてスケルトンライブラリを構成し，現実の問題に対して並列プログラムが効率的に構成できることを具体的に示した．他の類似研究に比べると，胡氏の研究結果は扱える並列データ構造の種類が多く，また，スケルトンとその組み合わせ機構が数学的により整理されており，系統的な展開が可能なことである．国際的にも高い評価を得ており，優れた研究成果をあげたと評価される．

7　主な論文等：

1. Kiminori Matsuzaki, Zhenjiang Hu, Kazuhiko Kakehi, Masato Takeichi, Systematic Derivation of Tree Contraction Algorithms, to apper in Parallel Processing Letters, 2005.
2. Hideya Iwasaki, Zhenjiang Hu, A New Parallel Skeleton for General Accumulative Computations, International Journal of Parallel Programming, 32 (5): 389-414, October 2004.
3. Tetsuo Yokoyama, Zhenjiang Hu, Masato Takeichi, Deterministic Second-order

Patterns, Information Processing Letters, Vol. 89, No.6, Elsevier, 2004. pp. 309-314.

4. **横山 哲郎**, **胡 振江**, **武市 正人**, **決定論的** 2 **階パターンとプログラム変換への応用**, **コンピュータソフトウエア**, 21 (5): 71-76, 2004.
5. Dana Na Xu, Siau-Cheng Khoo, Zhenjiang Hu, PType System : A Featherweight Parallelizability Detector, Second ASIAN Symposium on Programming Languages and Systems (APLAS 2004), Taipei, Taiwan, November 4-6, 2004. LNCS 3302, Springer Verlag. pp.197-212.
6. Kiminori Matsuzaki, Kazuhiko Kakehi, Hideya Iwasaki, Zhenjiang Hu, Yoshiki Akashi, A Fusion-Embedded Skeleton Library, International Conference on Parallel and Distributed Computing (EuroPar 2004), Pisa, Italy, 31st August - 3rd September, 2004. LNCS 3149, Spinger Verlag. pp.644-653 .
7. Kiminori Matsuzaki, Zhenjiang Hu, Kazuhiko Kakehi, Masato Takeichi, Systematic Derivation of Tree Contraction Algorithms, 4th International Workshop on Constructive Methods for Parallel Programming (CMPP 2004), Stirling, Scotland, UK, 14 July, 2004. pp. 109-124.
8. Kazuhiko Kakehi, Zhenjiang Hu, Masato Takeichi, List Homomorphism with Accumulation, 4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03), Lubeck, Germany. October 16-18, 2003. pp. 250-259.
9. Mizuhito Ogawa, Zhenjiang Hu, Isao Sasano, Iterative-free Program Analysis, 8th ACM SIGPLAN International Conference on Functional Programming, (ICFP 2003), Uppsala, Sweden: 25-29 August 2003. ACM Press. pp.111-123.
10. Tetsuo Yokoyama, Zhenjiang Hu, Masato Takeichi, Deterministic Second-order Patterns and Its Application to Program Transformation, International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR 2003) Uppsala, Sweden: 25-27 August 2003. pp. 165-178. Revised version appears in LNCS 3018, 2004. Springer Verlag. pp. 128-142
11. Kiminori Matsuzaki, Zhenjiang Hu, Masato Takeichi, Parallelization with Tree Skeletons, International Conference on Parallel and Distributed Computing (Euro-Par 2003), Klagenfurt, Austria, 26th - 29th August 2003. Lecture Notes in Computer Science 2790, Springer Verlag. pp.789-798. An extended version appears as Technical Report METR 2003-21, Department of Mathematical Informatics, University of Tokyo, 2003.
12. Zhenjiang Hu, Tomonari Takahashi, Hideya Iwasaki, Masato Takeichi, Segmented Diffusion Theorem (invited paper), 2002 IEEE International Conference on Systems, Man and Cybernetics (SMC 02), Hammamet, Tunisia, October 6-9, 2002. IEEE Press.
13. Wei-Ngan Chin, Zhenjiang Hu, Towards a Modular Program Derivation via Fusion and Tupling, The First ACM SIGPLAN Conference on Generators and Components (GCSE/SAIG 2002), Pittsburgh, PA, USA, October 6-8, 2002. Affiliated with (PLI 2002). Lecture Notes in Computer Science 2487, Springer Verlag. pp.140-155.
14. Zhenjiang Hu, Hideya Iwasaki, Masato Takeichi, An Accumulative Parallel Skeleton for All , 11th European Symposium on Programming (ESOP 2002), Grenoble, France, April 8 - 10, 2002. Lecture Notes in Computer Science 2305, Springer Verlag. pp.83-97.
15. **横山 哲郎**, **篠埜 功**, **胡 振江**, **武市 正人**, **変換戦略の記述に基づくプログラムの自**

動生成システムの実装, **情報処理学会論文誌**, Vol. 43, No. SIG 3(PRO 14), pp. 62-77, March 2002.