Data Flow Analysis used in Various Partial Redundancy Elimination Techniques

Aneesha N
M. Tech Student, Dept. of CSE
MES College of Engineering
Kuttippuram, Kerala, India

Sivananaintha Perumal
Assistant Professor, Dept. of CSE
MES College of Engineering
Kuttippuram, Kerala, India

ABSTRACT
Partial Redundancy Elimination (PRE) is a redundancy elimination transformation technique used in optimizing compilers to improve the program efficiency. The major benefit of PRE is that it can be extended to perform other optimizations like strength reduction, global value numbering at the same time. The effectiveness and the generality make PRE one of the most important optimizations in optimizing compilers. In this paper, an attempt has been made to summarize various classic and speculative PRE systems and data flow analysis used for getting the flow information to remove partially redundant computations in a program.

General Terms
Compiler Optimization.

Keywords
Data Flow Analysis, Optimizing Compiler, Classic and Speculative PRE systems, Strength Reduction, Global Value Numbering.

1. INTRODUCTION
The emergence of high level programming language has caused an increase in the distance between the source code and object code of programs. In order to improve the efficiency, it is desirable to increase the power of transformations for doing the global optimization that are performed by compilers. Redundancy elimination strategies are used to identify situations where a computation has already been performed within a certain context and can be replaced by a copy of the previous result. Code motion is used to improve the efficiency of a program by avoiding unnecessary re-computations of a value at run time.

Partial redundancy elimination (PRE) is an important redundancy elimination method first developed by Morel E and Renvoise C [Morel E et al. 1978] which eliminates expressions that are partially redundant in a program. This concept was introduced more than 30 years ago and in the last 20 years there have been successful attempts in developing efficient PRE systems. PRE inserts and deletes computations in the flow graph in such a way that after the transformation each path contains no more and generally fewer occurrences of any such computation than before. The PRE systems use a properly performed flow analysis to determine the cases where the optimization do apply and whether it is safe or not. Data flow analysis gives the information about the global manipulation of data. The PRE algorithms use various data flow analyses to get the flow information and these are modeled as bit vector problems. In order to preserve the semantics of the original program, insertions of computations corresponding to transformation must be safe. By targeting the elimination of partially redundant computations in a program, PRE systems automatically remove global common sub expressions and move invariant computations out of the loop.

PRE technique is used to eliminate the partially redundant expressions in a program. It performs loop invariant movement and common sub expression elimination together. An expression is a partially redundant one if the value computed by that expression is available on some path, but not all paths through the program to that expression. PRE can eliminate partially redundant expressions by inserting the partially redundant expression on the paths that do not already compute it, thereby making the partially redundant expression fully redundant. In Fig 1 [V K Paleri et al. 1998], expression a+b at block 4 is partially redundant. Fig 2. [V K Paleri et al. 1998], shows the elimination of partially redundant expression a+b.

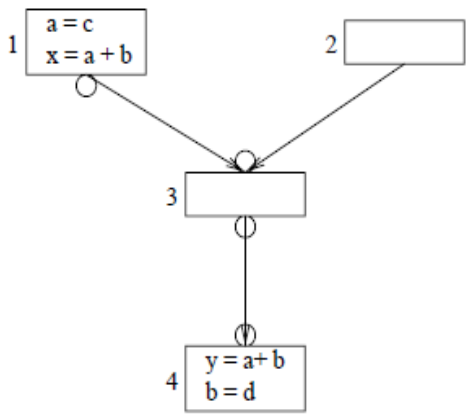

Fig 1: Partial Redundancy: Before Transformation

PRE systems consist of classic PRE systems and speculative PRE systems. Classic PRE systems are implemented as profile independent optimization technique. Traditionally PRE algorithms are implemented as classic PRE systems. In these systems the computation is placed at a point only if that point is safe for computation.

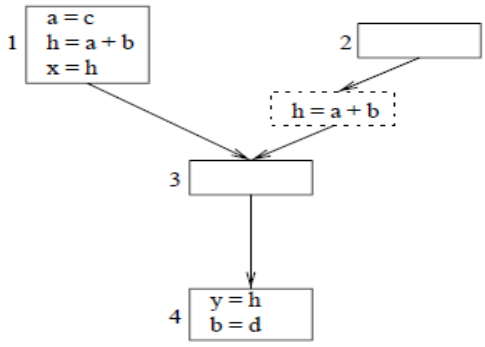

Fig 2: Partial Redundancy: After Transformation

The speculative PRE systems are implemented as profile dependent optimization technique. In profile dependent systems, it is possible to insert a computation at a point even if the computation is partially anticipable at that point. These systems use execution frequencies to find optimal places in a program to perform computations. Most of the PRE algorithms are optimal in nature and this optimal strategy eliminates the maximum number of redundant computations over all static execution paths without introducing any additional computations that would not be performed by the original program.

Usually, PRE is applied to lexically equivalent expressions, but Static Single Assignment (SSA) form based PRE systems also exist. In bit vector based PRE systems a large amount of time needs to be spent in analyzing the contents of each basic block to initialize the local data flow attributes which serves as the input to the data flow equations. PRE systems based on static single assignment form have benefits over other bit vector based PRE systems. It does not require collecting the data flow attributes over the program.

This paper gives an overview of various classic and speculative PRE systems and data flow analysis used for getting the flow information to remove the partially redundant computations in a program.

The rest of this paper is organized as follows. Section II surveys the various PRE systems aimed at improving the efficiency of PRE, Section III briefly discuss the analysis methods used in various PRE systems and Section IV concludes the survey.

2. VARIOUS PRE SYSTEMS

PRE is a powerful optimization technique used to eliminate the partially redundant expressions in a program by performing loop invariant movement and common sub expression elimination together. PRE can eliminate partially redundant expressions by inserting the partially redundant expression on the paths that do not already compute it, thereby making the partially redundant expression fully redundant. PRE was originally proposed by Morel E and Renvoise C [Morel E et al. 1978]. They proposed a technique for partial redundancy elimination in which PRE is treated as a bit vector problem. A strong bi directional iterative data flow analysis is performed to solve the bit vector problem. This method used the global analyses availability, anticipability and partial availability to get the data flow information. This technique is based on Boolean approach and the systems of Boolean equations are solved by direct iterative method. Here the elimination of redundant computations and movement of invariant computations are done at once and allowed to move each expression directly to the entrance of the outer most loops in which it is invariant.

This method is based on the data flow and not based on the control flow. Even if the algorithm introduced is efficient in time and space, it is not an optimal one and not eliminating all partial redundancies that exist in a program. The redundant code motion is another problem. The node placement is based on the bidirectional data flow equations, which is more costly to compute and more complex than unidirectional ones.

A modification to the algorithm proposed by Morel E and Renvoise C [Morel E et al. 1978] is done by the suppression of partial redundancies [Dhamdhere D.M 1991]. The presented technique is also treated PRE as a bit vector problem. It modified the concepts of local availability, local anticipability and transparency. A weak bi directional analysis is required to solve the problem. To inhibit redundant code motion, this method extends the edge placement technique [Dhamdhere D. M 1988]. Since the PPIN property has only a disjunctive dependence on the predecessor blocks, the Boolean properties Placement Possible on entry of a block (PPIN) and placement possible out (PPOUT) of this PRE algorithm represent a weak bidirectional data flow and the register pressure were partially addressed. The bound on the complexity of a weak bidirectional data flow is the same as that for a unidirectional data flow. So it is not more complex than Morel E and Renvoise C‘s PRE method [Morel E et al. 1978]. By taking the safety concept into account, this technique succeeded to remove all partial redundancies using the concept of edge placement. By using this method, it is not possible to avoid redundant code motion in all situations.

Lazy Code Motion (LCM) is an optimal bit vector algorithm for the elimination of partial redundancies [Knoop.J et al. 1992]. This algorithm decomposed the bidirectional structure of the PRE into a sequence of unidirectional analyses. The important feature of this algorithm is its laziness: computations are placed as early as necessary but as late as possible. It moves the computation even if there is no run time gain. Here first the critical edges in the flow graph are identified and then split them. To form the data flow analysis, various local and global data flow properties like transparency, local and global anticipability, earliness, delayness, latestness and isolated property. The introduced computationally optimal lazy code motion algorithm avoided the unnecessary register pressure and it is conceptually simple .The laziness guarantees the lifetime optimality while preserving computational optimality. This algorithm provides solution for only the single node single statement. It is more complicated to formulate, understand and solve.

A practice-oriented adaptation of the computationally and lifetime optimal code motion algorithm of lazy code motion methods consisted of unidirectional standard analyses and it worked on flow graphs whose nodes are basic blocks [Knoop.J et al. 1994]. It also considered the adaptation of as early as necessary but as late as possible computation placement strategy. To get the optimality result the computations are placed as early as possible, while maintaining the safety. In order to avoid the unnecessary code motion, the computations must be placed as late as possible while maintaining the computational optimality. Each step requires only two unidirectional data flow analyses. To form the data flow analysis, various local and global data flow properties like transparency, local and global anticipability, earliness, delayness, latestness and isolated property. Compared with the Lazy code motion method, the earliness

and isolation analyses are modified to decouple the analyses at each step, which then become independent and therefore parallelizable.

This algorithm is conceptually simple. It minimizes the number of computations in the program while suppressing any unnecessary code motion in order to avoid the superfluous register pressure. It is more complicated to formulate, understand and solve.

A computationally and lifetime optimal algorithm for partial redundancy elimination is the simple algorithm for PRE [V K Paleri et al. 1998]. This simple algorithm for PRE requires four unidirectional data flow analyses and is based on the concepts availability, anticipability, partial availability and partial anticipability. The main feature of this algorithm is that safety constraint is added to definition of partial availability and partial anticipability to introduce the concepts safe partial availability, safe partial anticipability and safe partial redundancy. This algorithm initially identifies the safe partially redundant computations and makes them total redundant by the insertion of new computations at proper points. The main advantage of this algorithm is its simplicity and it requires only four unidirectional bit vector analyses. Unlike LCM [Knoop.J et al. 1992], this algorithm not requires the edge splitting transformation. This is a computationally and lifetime optimal algorithm. This algorithm provides solution for only the single node single statement.

PRE can also be used to eliminate redundant register loads and stores introduced by spilling [Dmitri Bronnikov 2004]. PRE is traditionally used to move partially redundant code up the control flow graph. Redundant register store elimination requires code motion in the opposite direction. The proposed formulation removes all partial redundancies. All partially available and locally anticipatable expressions are removed by definition. Here the anticipability concept has a small change: It does not propagate anticipability up the control flow graph further than the entry to a block which computes the expression even if the block is transparent. This algorithm is so simple. PRE transformation is correct because none of the properties ever cross a block unless the block is transparent. The introduced formulation removes almost all partial redundancies. All partially available and locally anticipatable expressions are removed by definition.

In order to ensure strict semantics, the safety constraint restricts the ability of optimization strategies to move redundant computations out of loops. To overcome this limitation, several approaches to speculative code motion have been proposed, which ignore the safety constraint. These techniques are costly for a just-in-time (JIT) compilation framework. To address this deficiency, a novel approach called Isothermal Speculative Partial Redundancy Elimination (ISPRE) is developed that uses a simple threshold to select which expressions to speculatively hoist [Bernhard Scholz et al. 2006]. It is an intra-procedural analysis algorithm or each procedure of a program will be transformed by ISPRE independently of the other procedures. This approach begins by assuming that speculatively computed results are available everywhere in highly active or hot regions of the control flow graph and dubbed isothermality due to the fact that it partitions the code into two sub graphs, hot and cold, and treats all code the same within each region. Here removability and necessity can be implemented as unidirectional analyses using bit-vector representations of sets of expressions. By using this approach, the results achieved are close to optimal in practice. But its analysis time is at least as good as current compiler techniques for code motion. It is a technique suitable for use in JIT compilers. This approach abandons a guarantee of optimality in favor of simplicity and speed of analysis. ISPRE inserts computations only at the boundary between hot and cold regions.

In order to overcome the difficulties of ISPRE, a new technique Selective Speculative Partial Redundancy Elimination (SSPRE) is presented [Nigel Horspool et al. 2010] This uses threshold to select which expressions to speculatively hoist. It identifies two thresholds that dictate which computations to consider and which of those to speculatively hoist. Here only a quarter of the code need be considered and analyzed to achieve the full benefit of the approach. Unlike previous works, this technique is a straightforward generalization of standard busy code motion and lazy code motion formulations. To formulate the SSPRE flow analysis, an edge oriented analysis is performed to avoid the complications of the critical edge problem. This approach simplifies the equations and reduces the number of results which need to be maintained over the control flow graph. SSPRE improves upon the ISPRE threshold technique by supporting both speculative and conservative strategies using only one set of analyses. Implementing a successful speculative strategy is often complicated.

All these PRE systems have modeled the problem as systems of data flow equations. Regardless of how efficiently the systems of data flow equations can be solved, a sustained amount of time needs to be spent in scanning the contents of each basic block in the program to initialize the local data flow attributes that serves as input to the data flow equations. It takes more time than solving the data flow equations. So another approach to PRE is needed which does not require the dense initialization of the data flow information. The optimizations based on the SSA form share some common characteristics and they do not require traditional iterative data flow analyses. Also they take the advantages of the sparse representation of SSA. The SSA based optimization algorithms can handle both local and global versions of the optimizations simultaneously. So it is possible to reduce the effort required to implement the optimization.

Locality based static single assignment PRE provides a practical improvement to the analysis time of the original Static Single Assignment Partial Redundancy Elimination algorithm (SSAPRE) [Jongsoo Park et al.2008]. The original SSAPRE algorithm identifies Φ assignments that are unsafe to insert new computation for redundancy elimination at a later stage of SSAPRE. Before applying SSAPRE, all critical edges in the control flow graph have been removed by inserting an empty basic block at each critical edge. The new locality based SSAPRE algorithm identifies most of unnecessary Φ assignments for local expressions with simple occurrence and post-dominance checks and filters out most of unnecessary Φ assignments at the first stage i.e. the Φ -Insertion phase, thus it reduces the analysis time in the following stages. SSAPRE has benefits over other bit-vector based PRE algorithms. It preserves the properties of the SSA form after PRE and exploits the sparsity of the SSA form, resulting in reduced analysis and optimization time.

3. ANALYSIS METHODS USED IN PRE SYSTEMS

The various PRE systems use different analysis method to collect the flow information. A large amount of time needs to be spent in analyzing the contents of each basic block in the program to initialize the local data flow attributes which serves as the input to the data flow equations.

In a strong bi directional data flow analysis [Morel E et al. 1978] the node properties strongly depend on its predecessors and successors. In weak bidirectional algorithm [Dhamdhere D.M 1991], the properties of a node strongly depend on its successors but weakly depend on its predecessors and thus the analysis is a weak bidirectional analysis. In both cases availability, anticipability and partial availability are used. The lazy code motion algorithm [Knoop.J et al. 1992], busy code motion [Knoop.J et al. 1994], simple algorithm for PRE [V K Paleri et al. 1998], Dmitri Bronnikov's PRE method [Dmitri Bronnikov 2004], Isothermal speculative PRE methods [Bernhard Scholz et al. 2006], and Selective Speculative PRE method [Nigel Horspool et al. 2010] use unidirectional data flow analysis. i.e. data flow information at node of the program flow graph is influenced either by its predecessor or by its successor. In SSAPRE systems [Jongsoo Park et al.2008] no iterative data flow analysis is required. This method performs the partial redundancy elimination based on the locality and dominance relationship of the expressions. So there is no need to collect the data flow attributes over the program.

4. CONCLUSION AND FUTURE WORK

The scope of this review paper is focused on various PRE systems and data flow analysis. The effectiveness and the generality make PRE one of the most important optimizations in optimizing compilers. There are various PRE systems are used in optimizing compilers. Regardless of how efficiently the systems of data flow equations can be solved, large amount of time needs to be spent in scanning the contents of the basic blocks. The time required to analyze the blocks are greater than the time required to solve the data flow equations. Since SSAPRE algorithm is based on dominance relationships, it does not require to collect data flow attributes and does not involve any iterative data flow analysis and bit vectors in its solution. Even though these SSAPRE systems have good performance, it contains unnecessary Φ assignments. The unnecessary Φ can be removed by modifying the SSAPRE algorithm and it will improve the performance of SSAPRE systems.

5. REFERENCES

[1] Morel E and Renvoise C. 1978. Global Optimization by Suppression of Partial Redundancies, In Communication of ACM.

[2] Dhamdhere D.M 1991. Practical adaptation of the global optimization algorithm of Morel and Renvoise, ACM Trans. on Programming Languages and System.

[3] Knoop.J, Ruthing O,Steffen B. 1992. Lazy code motion , ACM SIGPLAN92 conference on Programming language design and implementation.

[4] Knoop J, Steffen B and Ruthing O. 1994. Optimal code motion: Theory and practice, In Proceedings of ACM Transactions programming language systems.

[5] V K Paleri, Srikant Y. N, Shankar P 1998. A Simple Algorithm for Partial Redundancy Elimination., In Proceedings of ACM SIGPLAN Notices.

[6] Dmitri Bronnikov. 2004. A Practical Adoption Of Partial Redundancy elimination, In Proceedings of ACM SIGPLAN.Med. Image Analysis.

[7] Bernhard Scholz ,R Nigel Horspool, David J Pereria. 2006. Fast Profile-Based Partial Redundancy Elimination, In Proceedings of 7th Joint Modular language Conference.

[8] R Nigel Horspool and Rhodes H. F. Brown 2010 Selective, Speculative Partial Redundancy Elimination, In Proceedings of International Conference on Compiler Construction .

[9] Jongsoo Park , Jaejin Lee. 2008. A Practical Improvement to the Partial Redundancy Elimination in SSA

[10] Dhamdhere D. M 1988. A fast algorithm for code movement optimization. In Proceedings of ACM SIGPLAN Notices .