Limits of Parallelism on Thread-Level Speculative Parallel Processing Architecture

Metsugi, Katsuhiko
Department of Informatics, Kyushu University

Murakami, Kazuaki
Department of Informatics, Kyushu University

出版情報：SLRC 論文データベース, 2002-05. International Workshop on Informationons and Electrical Engineering
バージョン：
権利関係：

LIMITS OF PARALLELISM ON THREAD-LEVEL SPECULATIVE PARALLEL PROCESSING ARCHITECTURE

Katsuhiko Metsugi and Kazuaki Murakami

Department of Informatics
Kyushu University
6-1 Kasuga, Fukuoka 816-8580, JAPAN

ABSTRACT

Two fundamental restrictions that limit the amount of instruction-level parallelism extracted from sequential programs are control flow and data flow. TLSP (Thread-Level Speculative Parallel processing) architecture gains high parallelism using three techniques (speculation with branch prediction, control dependence analysis, executing multiple flows of control) which relax constraints due to control dependences. In this paper, we evaluate the effects of three techniques (memory disambiguation, renaming, value prediction) which relax constraints due to data dependences on TLSP architecture. We have two major results. First, parallelism for TLSP architecture is restricted by enormous output and anti dependences on memory. Second, value prediciton has large effects on TLSP architecture.

1. INTRODUCTION

There are two fundamental restrictions that limit the amount of parallelism that can be extracted from sequential programs: control flow and data flow. Control flow limits parallelism by imposing serialization constraints at forks and joins in a program's control flow graph. Data flow limits parallelism by imposing serialization constraints on pairs of instructions that are data dependent (i.e., one needs the result of another to compute its own result, and hence must wait for the other to complete before beginning to execute). Examining the extent and effect of these limits has been a popular and important area of research, particularly in the case of control flow.

Wall's work[15] explained the parallelism obtained from sequential program with branch prediction, register renaming and alias analysis. The result parallelism of Wall's work was surprisingly small and this result incited Lam and Wilson's work[7]. Lam and Wilson's work extended the parallelism by relaxing control flow constraints between instructions from the following three points of view.

1. Speculation with branch prediction
2. Control dependence analysis
3. Executing multiple flows of control

Combination of these techniques especially introduces large parallelism and triggers many architectures called "TLSP (Thread-Level Speculative Parallel processing) architecture". The following TLSP architectures have been proposed so far.

- Multiscalar[13] at UW-Madison
- MUSCAT[12], thereafter MP98[10] at NEC
- Hydra[3] at Stanford
- Speculative CMP(Chip MultiProcessor)[6] at UIUC
- TLDS(Thread-Level Data Speculation)[14] at CMU
- OCHA-Pro[11] at Tokyo University
- SKY[5] at Nagoya University

However, Lam and Wilson's work and researches on TLSP architectures have three problems stated below.

1. Although there is conspicuous disjunction of parallelism between Lam and Wilson's work and the performance evaluation of TLSP architectures shown in Table 1 and 2, there are few considerations about the reason.
2. Lam and Wilson's work defines one manner to handle data dependences, assuming that only true dependences are kept wherever operands exist and that dependences of memory operands are known beforehand. Although this is an ideal approach to handle data dependences, data dependences contains false data dependences, such as anti-dependence and output dependence, and therefore the way to handle these dependences affects parallelism. And again, we have a problem, called "memory ambiguity", that means that we cannot know the address of data dependences on memory until execution. After all, Lam and Wilson's work is effective to know the limit of parallelism obtained by using the way to relax constraints

Table 1. Results of Lam and Wilson's Evaluation

		awk	ccom	eqntott	espresso	gcc	irsim	latex	Harmonic Mean
Lam and Wilson[7] ⟨Parallelism⟩	BASE	2.85	2.13	1.98	1.51	2.10	2.31	2.71	2.14
	CD	3.24	2.51	2.05	1.54	2.55	2.66	3.17	2.39
	CD-MF	5.32	5.61	5.21	7.49	14.63	11.89	6.18	6.96
	SP	9.22	6.92	6.40	4.16	7.76	8.40	7.60	6.80
	SP-CD	12.89	9.83	18.09	19.55	13.18	15.82	9.72	13.27
	SP-CD-MF	41.88	18.05	225.90	402.85	66.29	45.86	18.65	39.62
	ORACLE	242.77	46.80	3282.91	742.30	174.50	265.42	131.69	158.26

Table 2. Comparison of ILP on TLSP Architectures

		compress	eqntott	espresso	gcc	go	idct	li	vortex
Multiscalar[13] ⟨UICR⟩	1PE	0.74	0.85	0.93	0.86				
	2PEs	1.52	1.32	1.44	1.25				
	4PEs	2.15	2.18	1.96	1.56				
	8PEs	2.44	2.39	2.98	1.70				
	12PEs	2.52	2.43	3.74	1.73				
MUSCAT[12] ⟨Speedup⟩	1PE	1.00	1.00				1.00		
	2PEs	1.15	1.18				1.98		
	4PEs	1.48	1.60				2.90		
	8PEs	1.80	2.18				3.00		
Hydra[3] ⟨Speedup⟩	4PEs	1.00	0.58					1.04	0.62
SKY[5] ⟨IPC⟩	2PEs	1.2			2.4	2.7		2.8	3.1
	4PEs	1.9			2.4	3.0		2.9	3.2
	8PEs	3.3			2.5	3.1		2.9	3.2

UICR: Useful Instruction Completion Rate

IPC: Instruction Per Clock cycle

Speedup: Speedup of nPEs over 1PE

due to control dependences, but it offers no information about the correlation between the ways to handle data dependences and control dependences.

3. Some TLSP architectures employ value prediction techniques to relax constraints due to true data dependences. These architectures are called TLDSP (Thread-Level Data-Speculative Parallel processing) architecture. Though value prediction is fairly strong way to relax constraints due to true data dependences, no evaluation about the effect have been performed.

On TLSP architectures, not only intra-thread data dependences but also inter-thread data dependences define parallelism. Therefore, we try to know what kind of data dependences have been underlying within and without threads by solving above-mentioned problem 2 and 3. To solve these problems, we summarize techniques to relax constraints due to control and data dependences and define abstract machine models which use those techniques to evaluate the effects. And we show the result parallelism obtained by using abstract machine models.

The rest of this paper is organized as follows: Section 2 explains the techniques to relax the control flow constraints. Section 3 explains the techniques to relax data flow constraints. Section 4 describes the experimental framework. Section 5 presents the results of these experiment and analyzes the results.

2. THREAD-LEVEL SPECULATIVE PARALLEL PROCESSING ARCHITECTURE

Lam and Wilson's work uses the following three techniques to relax constraints due to control dependences.

1. **Speculation with Branch Prediction:**
   If a branch instruction exists, successive instructions must wait execution until the branch instruction is resolved. However, it is able to predict the direction of

the branch instruction and execute the successive instructions. If the prediction succeeds, the constraint due to the branch instruction can be relaxed.

2. **Control Dependence Analysis:**
   When we note a branch instruction, not all of the following instructions are control dependent on the branch instruction. Then if we can analyze control dependence between individual instructions precisely, instructions that have no control dependence on the branch instruction can execute independently of the branch instruction.

3. **Executing Multiple Flows of Control:**
   In a program, multiple flows of control can exist. If multiple processors can execute multiple flows of control, constraints due to control dependences can be relaxed.

We call the architecture using all above techniques as "TLSP architecture". The result of Lam and Wison's work shown in Table 1 tells us that 'Speculation with Branch Prediction" and "Executing Multiple Flows of Control" introduce enormous parallelism.

In Lam and Wilson's work, TLSP architecture is called "SP-CD-MF" and the definition is below.

SP-CD-MF An instruction cannot execute until its mispredicted control dependence branches are resolved. There are no additional constraints on branches.

To evaluate TLSP architecture, we also use above abstract machine model.

3. RELAXING DATA DEPENDENCE CONSTRAINTS

Data dependences are classified into three types: flow dependence, anti-dependence and output dependence. In addition, the way to relax constraints due to these data dependences are independent from each other. In this paper, we consider the following techniques.

1. **Memory Disambiguation:**
   Generally, we cannot know memory reference addresses before execution. In this case, we cannot know dependences between instructions which accesses memory until its execution. If we can analyze dependences between memory access instructions by means of some methods, constraints due to memory ambiguation can be relaxed.

2. **Renaming:**
   Anti-dependences and output dependences are false dependences caused by reusing resources (registers and memories). False dependences, differently from true dependences, can be eliminated by renaming.

3. **Value Prediction:**
   Flow dependences are true dependences where an instruction cannot execute until all flow dependent instructions are finished. However, if the value of the data which causes flow dependence can be predicted, flow dependent instruction can execute speculatively using the predicted value. If the value prediction is correct, constraints due to flow dependences can be relaxed.

Same as control dependence constraints, we define the following abstract machine models regarding data dependences.

dBASE: An instruction cannot execute until all preceding data dependent instructions finish. In addition, memory reference instruction cannot execute until all preceding memory reference instructions finish their execution.

MD: An instruction cannot execute until all preceding data dependent instructions finish.

RN: An instruction cannot execute until all preceding flow dependent instructions finish. In addition, memory reference instruction cannot execute until all preceding memory reference instructions finish their execution.

dSP: A mispredicted instruction cannot execute until all preceding data dependent instructions finish. A successfully predicted instruction cannot execute until all preceding anti and output dependent instructions finish. In addition, memory reference instruction cannot execute until all preceding memory reference instructions finish their execution.

RN+dSP: An mispredicted instruction cannot execute until all preceding flow dependent instructions finish. In addition, a memory reference instruction cannot execute until all preceding memory reference instructions finish their execution.

MD+dSP: A mispredicted instruction cannot execute until all preceding data dependent instructions finish. A successfully predicted instruction cannot execute until all preceding anti and output dependent instructions finish.

MD+RN: A instruction cannot execute until all preceding flow dependent instructions finish.

MD+RN+dSP: An mispredicted instruction cannot execute until all preceding flow dependent instructions finish.

dORACLE: There are no constraints due to data flow.

Table 3. Benchmark Programs

Program	Description
gzip	Compression
vpr	FPGA Circuit Placement and Routing
mcf	Combinatorial Optimization
parser	Word Processing
perlbmk	PERL Programming Language
vortex	Object-oriented Database

4. EXPERIMENTAL FRAMEWORK

To know the effects of relaxing constraints due to data dependences on TLSP architecture, we do trace based simulation using abstract machine models defined Section 2 and 3. We use the same simulator used in Lam and Wilson's work and modify it to implement our new machine models. Therefore, the basic policy of evaluation conforms to Lam and Wilson's work.

4.1. Program Trace

Our simulator is trace-driven. Trace includes instruction and basic block information at the time of execution. Though traces are made by MIPS pixie tool in Lam and Wilson's work, we use SimpleScalar[1] toolset to make traces. So that our simulator is modified to interpret SimpleScalar instruction set.

4.2. Program Transformations

Procedure calls and loops introduce unnecessarily serializing constraints. These constraints are unnecessary for our evaluation because our aim is to investigate the limits of parallelism. So we do procedure inlining and loop unrolling to eliminate such serializing constraints. In our simulator, we ignore all procedure call and return instruction in a trace, as well as all instruction that manipulate the stack pointer. In addition, we also ignore all instruction concerning about loop index variables and branch instruction to do with loop.

4.3. Benchmark Programs

We use six programs from SPEC CPU2000 benchmark suit shown in Table 3. Since we use SimpleScalar toolset, the benchmark programs are compiled for the SimpleScalar architecture.

4.4. Simulation Algorithm

Determining the execution timing of each instruction in the trace is the basic simulation algorithm. Execution timing follows corresponding abstract machine models defined Section 2 and 3. Basically we follow Lam and Wilson's algorithm which records the time of the most recent write to each register and memory location. Since we define nine models to handle data dependences, a different manner, which simulate abstract machine models defined in Section 3, is required to record the time of the most recent write. Total execution time of the trace equals the completion time of the last instruction to execute. We assume that each instruction in the trace has one clock cycle latency. The reported result parallelism is instruction-level parallelism.

4.5. Branch Prediction Method

For TLSP architecture, branch prediction method is very important. Unlike Superscalar processor, when branch instruction is executed on TLSP architecture, there is no guarantee that all preceding branch instructions have finished. So we cannot simply use modern branch prediction method which exploits information of preceding branch instructions on TLSP architecture. To solve this problem, we use hybrid branch prediction method which does gshare[9] branch prediction and static branch prediction based on profile information. If all preceding branch instructions are finished before an branch instruction executed, we use gshare branch prediction, otherwise we use static branch prediction.

Our gshare branch predictor provide 128K entries of pattern history table and 12 history depth of branch history register.

4.6. Value Prediction Method

Machine models which include dSP need a concrete value prediction mechanism. Some value prediction mechanisms have been proposed so far. We choose a stride value predictor because this is general and easy to implement. Our stride value predictor has 4096 value history table entries.

5. RESULTS

The parallelism for each abstract machine models are shown in Table 4 and Figure 5.

dBASE machine provides a standard for comparison by determining the amount of parallelism when no effort is made to relax constraints due to data dependences. dBASE machine has a harmonic mean parallelism of 3.83.

dORACLE machine provides upper bound of TLSP architecture. Harmonic mean parallelism is 49.06.

5.1. Effect of Memory Disambiguation

Harmonic mean parallelism for MD machine is 4.58 and is much lower than other machine models like RN or dSP. But Parallelism for parser and perlbmk are higher than RN.

Table 4. Parallelism for each Machine Model of TLSP

	gzip	vpr	mcf	parser	perlbmk	vortex	Harmonic Mean
dBASE	2.16	4.67	4.09	6.60	3.97	4.16	3.83
MD	2.80	5.07	4.67	8.35	4.91	4.56	4.58
RN	3.72	9.43	7.35	7.60	4.61	8.16	6.11
dSP	5.71	6.66	4.09	12.73	8.37	8.32	6.76
MD+dSP	9.01	13.28	7.35	15.16	10.04	11.14	10.45
RN+dSP	6.14	8.34	4.67	14.71	10.24	12.51	8.08
MD+RN	35.20	53.64	30.58	15.58	11.24	45.99	23.77
MD+RN+dSP	35.36	80.84	31.60	24.99	29.04	68.57	37.27
dORACLE	35.38	89.27	79.00	25.25	43.76	126.52	49.06

5.2. Effect of Renaming

Harmonic mean parallelism for RN is higher than MD but lower than dSP. Parallelism for some programs like vpr and mcf are higher than dSP.

Without MD, RN's renaming is limited on registers. When we use MD+RN, not only constraints of sequential access to memory is relaxed, but also renaming of memory access is done. If Figure5 is seen, it is quite obvious that TLSP architecture is characterized by MD+RN. Harmonic mean parallelism for MD+RN is 23.77, which is lower than SP-CD-MF of Lam and Wilson's work in Table 1. The difference comes from the manner of potential data dependences arise from control flow. Lam and Wilson's work ignore potential data dependences and our work do not ignore.

5.3. Effect of Value Prediction

Irrespective of other techniques, the effect of value prediction is quite large on TLSP. Harmonic mean parallelism for dSP machine is 6.76 and this is higher than MD+RN. But parallelism for some programs like vpr and mcf are lower than RN machine.

Combinatorial use of dSP and other single technique is also effective. Harmonic mean parallelism for MD+dSP machine is 10.45 and is 128% increase as compared with MD machine. But compared with MD+RN machine, parallelism is much lower.

When we use value prediction on MD+RN which is the general format of TLSP, harmonic mean parallelism is 37.27 and this is 57% increase from MD+RN. If all value prediction have succeed, harmonic mean parallelism is 49.06 and this is 106% increase of parallelism from MD+RN.

6. CONCLUSIONS

This paper shows that what kind of data dependences are underlying in TLSP architecture and how we can relax it. Through this study, we find that the parallelism of TLSP architecture is restricted by enormous output and anti dependences on memory. Furthermore, value prediction is very effective in relaxing flow dependences of TLSP architecture. But we also find that perfect value prediction can obtain twice as more parallelism on TLSP architecture.

As future work, we would solve residual problem of TLSP represented in Section1. To do so, we should impose some restriction on machine models. By doing so, we can do more precise comparison to establish the method for determining the use of techniques relaxing constraints due to data and control dependences.

7. REFERENCES

[1] Burger, D. and Austin, T. M., "The SimpleScalar Tool Set, Version 2.0," *University of Wisconsin-Madison Computer Sciences Department Technical Report*, #1342, June 1997.

[2] Dubey, P. K., O'Brien. K., O'Brien, K. M. and Barton, C.,"Single-Program Speculative Multithreading (SPSM) Architecture: Compiler-assisted Fine-Grained Multithreading," *1st International Conference on Parallel Architectures and Compilation Techniques*, pp.109–121, June 1995.

[3] Hammond, L., Hubbert, B., Siu, M., Prabhu, M., Chen, M. and Olukotun, K., "The Stanford Hydra CMP," *IEEE MICRO Magazine*, pp.71–84, March-April 2000.

[4] Kemp, G. A. and Franklin, M., "PEWs: A Decentralized Dynamic Scheduler for ILP Processing," *International Conference on Parallel Processing*, Vol.1, pp.239–246, August 1996.

[5] Kobayashi, R., Iwata, M., Ogawa, Y., Ando, H. and Shimada, T., "An On-Chip Multiprocessor Architecture with a Non-Blocking Synchronization Mechanism," *25th EUROMICRO Conference*, pp.432–440, September 1999.

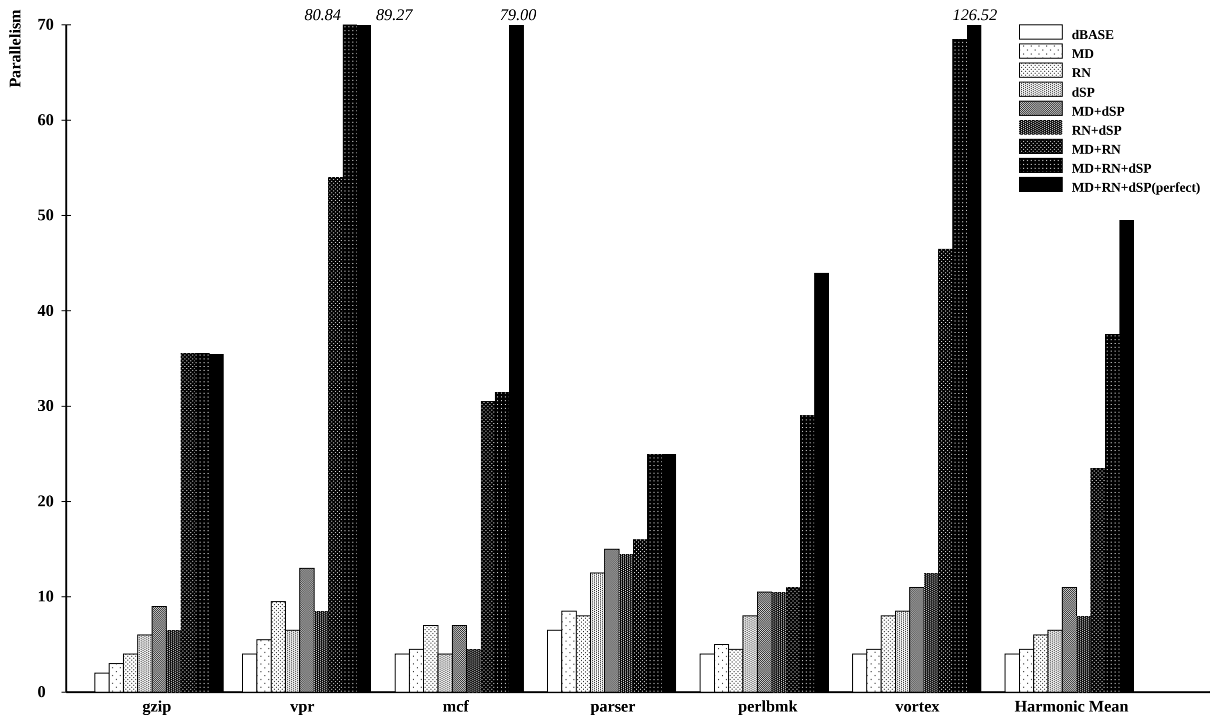

Fig. 1. Parallelism for each Machine Model of TLSP

[6] Krishnan, V. and Torrellas, J., "Hardware and Software Support for Speculative Execution of Sequential Binaries on a Chip-Multiprocessor," *12th International Conference on Supercomputing*, July 1998.

[7] Lam, M. S. and Wilson, R. P., "Limits of Control Flow on Parallelism," *19th International Symposium on Computer Architecture*, pp.46–57, June 1992.

[8] Lipasti, M. H. and Shen, J. P., "Exceeding the Dataflow Limit via Value prediction," *29th International Symposium on Microarchitecture*, pp.226-237, 1996.

[9] McFarling, S., "Combining Branch Predictors," *WRL Technical Note*, TN–36, June 1993.

[10] Nishi, N., Inoue, T., Nomura, M., Matsushita, S., Torii, S., Shibayama, A., Sakai, J., Ohsawa, T., Nakamura, Y., Shimada, S., Ito, Y., Edahiro, M., Minami, K., Matsuo, O., Inoue, H., Manabe, T., Horiuchi, T., Motomura, M., Yamashina, M. and Fukuma, M., "A 1GIPS 1W Single-Chip Tightly-Coupled Four-Way Multiprocessor with Architecture Support for Multiple Control Flow Execution," *International Solid-State Circuits Conference*, February 2000.

[11] Tamatsukuri, J., Matsumoto, T. and Hiraki, K., "Large-scale speculative parallel execution mechanism on On-Chip MIMD (in Japanese)," *IPSJ SIG Notes*, ARC–125–24, August 1997.

[12] Torii, S., Kondo. M., Motomura, M., Ikego, A., Konagaya, A. and Nishi, N., "On-chip Control Parallel Multi-processor: MUSCAT (in Japanese)," *Trans. of Information Processing Society of Japan*, Vol.39, No.6, June 1998.

[13] Sohi, G. S., Breach, S. E. and Vijaykumar, T. N., "Multiscalar Processors," *22th International Symposium on Computer Architecture*, pp.414–425, June 1995.

[14] Steffan, J. G. and Mowry, T. C., "The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelizetion," *4th International Symposium on High-Performance Computer Architecture*, January 1998.

[15] Wall, D. W., "Limits of Instruction-Level Parallelism," *Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp.177-188, April 1991.