⑪A Speculation Technique for Loops Containing Loop Carried Dependence

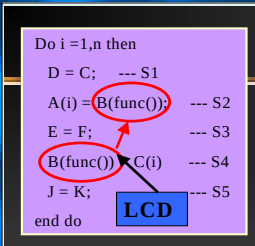
Shunsuke ISHIKAWA[1,2], Fumiko SAITO[1,2], Hayato YAMANA[1,2]
([1] APC Technology Group [2]Waseda University)

Contents

Introduction

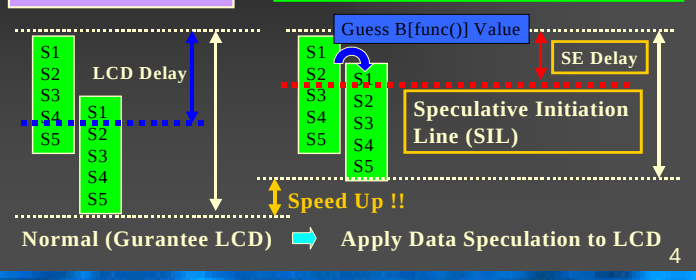
- A number of loop parallelization techniques have been proposed. However, there is no effective technique to parallelize loops contain **Loop Carried Dependence(LCD)**.
- Generally, the speculative execution techniques parallelize these loops.
- Purpose of this presentation is to propose a speedup technique to speculate loops containing LCD.

Overview of the Proposed Technique



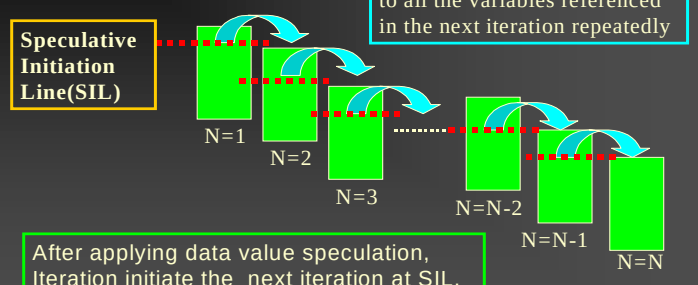
To speedup execution time,

1. Apply the Data Speculation to B[func()] causing LoopCarriedDependence (LCD) .
2. Initiate the next iteration quickly.



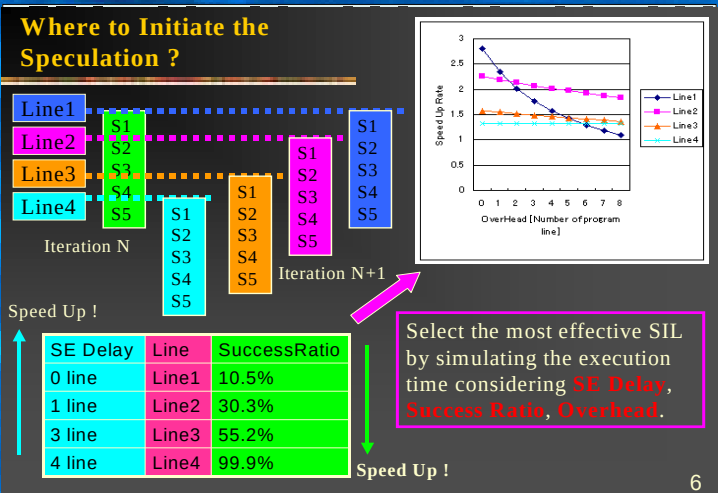
Overview of the Proposed Technique (2)

Apply data value speculation to all the variables referenced in the next iteration repeatedly



After applying data value speculation, Iteration initiate the next iteration at SIL. This execution model is like the "DoAcross" execution model.

Where to Initiate the Speculation ?



SE Delay	Line	SuccessRatio
0 line	Line1	10.5%
1 line	Line2	30.3%
3 line	Line3	55.2%
4 line	Line4	99.9%

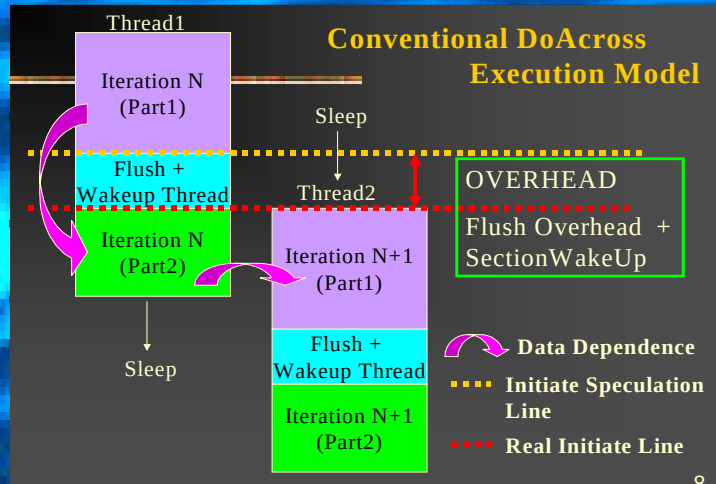
Select the most effective SIL by simulating the execution time considering SE Delay, Success Ratio, Overhead.

Speed Up !

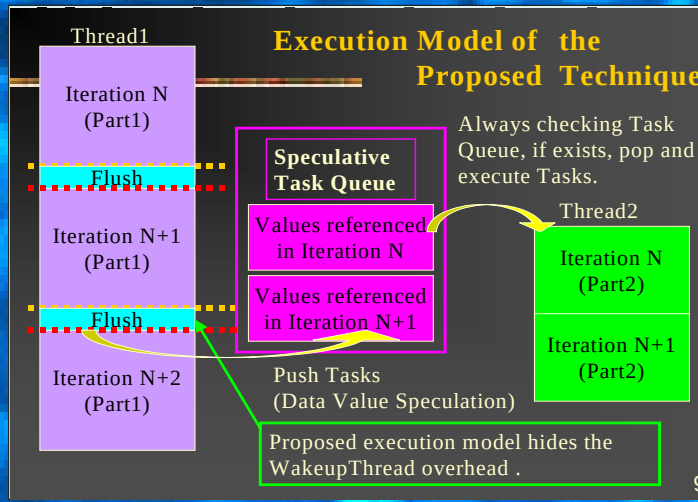
How to Calculate the Success Ratio ?

- List the all parameters required in calculating the Speculation Success Ratio.
  - Execution probability of every control flow.
  - Success Ratio of data value speculation to the variables causing LCD for every control flow.
    - Data value speculation schemes are LastValue, StrideValue.
- All required parameters are detected by the REX System automatically.

Conventional DoAcross Execution Model



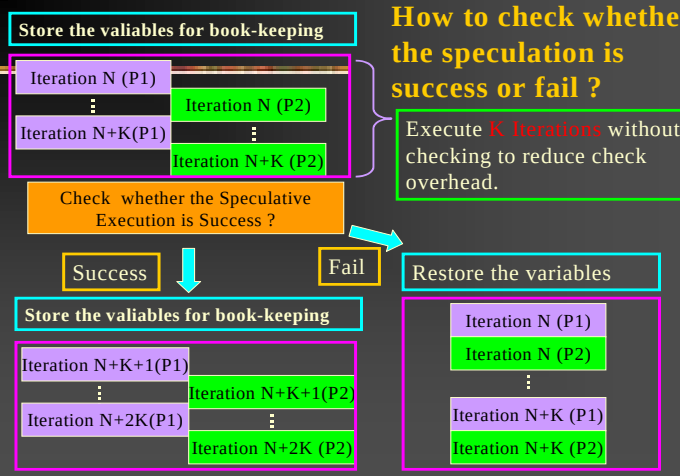
Execution Model of the Proposed Technique



Always checking Task Queue, if exists, pop and execute Tasks.

Proposed execution model hides the WakeupThread overhead .

How to check whether the speculation is success or fail ?

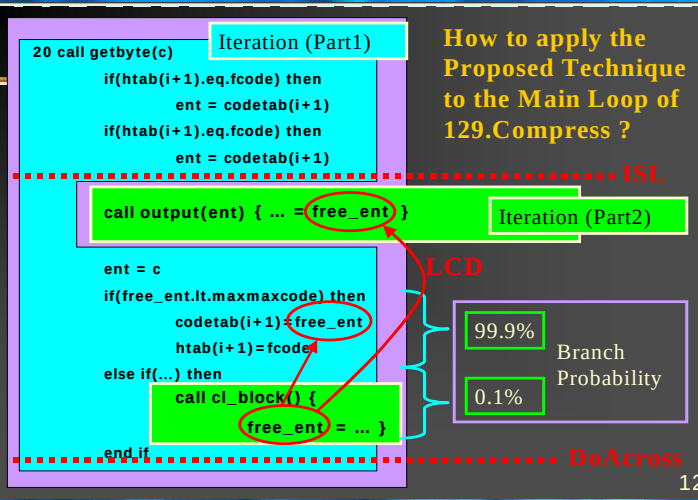


Execute K Iterations without checking to reduce check overhead.

Evaluation

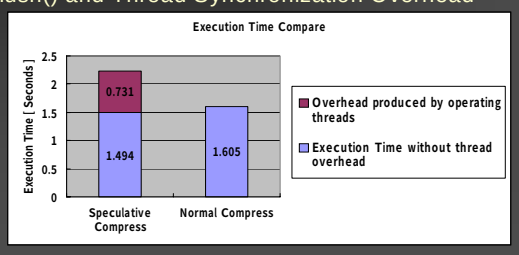
- Confirm the Proposed Technique Efficiency by applying to 129.compress (SPEC CINT95)
- Environments
  - Machine : IBM pSeries690 RegttaH
    - Power4(1.1GHz) ✕8 (16Processors)
  - Compiler : xlf_r -O5 –qsmp=omp
  - Parallelize Directive : OpenMP 1.1
  - Environment Variables :
    - AIX_THREAD_SCOPE=S
    - SPINLOOPTIME=0:YIELDLOOPTIME=0

How to apply the Proposed Technique to the Main Loop of 129.Compress ?



Comparison of the Execution Time

- The overhead produced by operating threads is much longer than the average iteration execution time of compress.
  - The initiation and the termination overhead of SECTION.
  - Flush() and Thread Synchronization Overhead



Our Understanding Resulted from Experimental Results

- If one Iteration average overhead produced by operating threads is larger than average execution time of one Iteration, Speculative Execution do not work effectively.

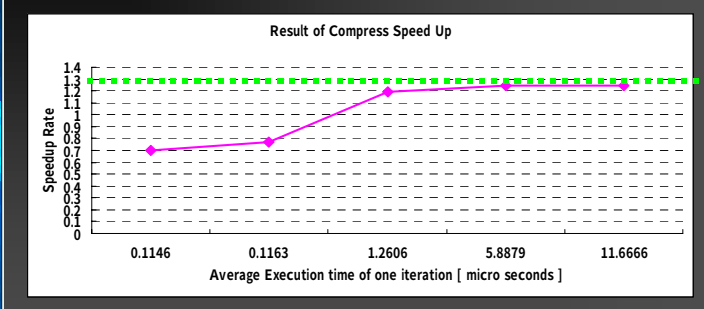
Average Execution Time of one iteration : T

Average Overhead produced by operation threads : O

Theoretical Max Speed Up Rate : R

→ **O is required to be T/R + O < T**

Confirmation of the Efficiency of Proposed Technique



Conclusion

- We propose the technique to apply the speculative execution alternatively only to the portion expected to be speeduped effectively, and confirm its effectiveness by applying the technique to existing benchmarks.
- The overhead produced by operating threads is very large. Architectural operating thread optimization is highly required.