Speech Synthesizer with Speculative Multithreading and Speculative Computation reuse

Suma S*, N.P. Gopalan**
*Department of MCA, PESIT, Bangalore, India
**Department of MCA, NIT, Trichy, India
Suma_bramara@yahoo.com, npgopalan@yahoo.com

Abstract:
Speculative multithreading and speculative computation reuse uses profiles for exploiting and to increase the cache memory performance. Currently proposed multithreaded processors try to improve on execution of number of instructions in each clock cycle. Even the general purpose applications can enchance the Instruction level parallelism through these techniques. Hence in this paper we are proposing an idea of the hybrid technique of both multithreading and instruction reuse to enhance the performance and execution rate comparatively with the sequential computation. We are also computing the thread interval for the reliability testing such that at what interval the value to be passed from the producer thread to the consumer thread.

Keywords: **speculative multithreading, speculative computation reuse, Instruction level parallelism, thread level parallelism, data prefetching, Reliability, spawned thread,**

I. Introduction:

The potential instruction-level parallelism exists in programs have to overcome control flow and data flow constraints that are inherent in parallelizing the programs. One can take the advantage of available abundance of the transistors on new computer architectures. Speculative parallelism has become the mainstream technique to exploit instruction-level and thread level parallelism. The developments in compiler technology[1,2].

Uses profiles to support data prefetching, thread level speculation, speculative computation reuse and speculative pre computation for increasing the performance of the cache memory.

Some of the compiler frameworks support data and control speculation for many compiler optimizations techniques and using thread level parallelism to enhance ILP speculatively for general purpose applications. Speculative computation reuse [1,4,5,6,7,8,9] uses value profile to help computation reuse. To avoid unnecessary invalidation speculative computation reuse uses speculative multithreading hardware. In Speculative reuse, there is no need to insert invalidation instructions. The speculative multithreading hardware is used to overlap the speculative execution of the code after the region with the execution of the computation region that verifies the predicted output. The speculative reuse will not commit result to memory or register file until the verification thread confirms that the predicted output is correct. If the verification thread finds that the reused output is incorrect, the speculative execution will be squashed. At that time, the output can be used in the code region is available from the region non-speculatively.

Experimental result shows that speculative computations reuse with 4 inputs/outputs per region can achieve up to 40% performance gain. [1,9]

Speculative parallel threading is a technique which selects good loops in the program directed by the compiler frameworks. In this paper we are proposing a novel idea of combining the speculative parallel threading with the speculative computation reuse so that the hybrid of the technique results in speculatively parallel reusable computation technique where the compiler frameworks select the part of the code which is a candidate for the loop processing and it is the loop that has to be executed repeatedly or reused repeatedly.

Using this knowledge we propose a technique which is called SPRC (speculatively parallel reusable computation). We propose an idea to prove this technique illustrating with speech/sound recognition problem.

978-1-4673-2818-0/13/$31.00 ©2013 IEEE

II Digraph Representing Speech:

Basically, Dynamic programming on a Digraph G (V,E) can be used for speech recognition. The edges (u,v)belongs to E labeled with a sound S(u,v) from a set of sounds. The labeled graph determines the person speaking a language. Each path in the graph starts from a distinguished vertex V0 belongs to V corresponds to a possible sequence of sounds produced by the model. The labels of the directed graph are defined to be the concatenation of labels edges on that path. Every edge (u,v) belongs E has also been given an associated nonnegative probability p(u,v) of traversing the edge(u,v). the probability of the path is defined as product of probabilities of the edges. We can view the probability of a path beginning at V0 as the probability that a random walk begin at V0 will follow the specified path, where the choice of which edge to select at a vertex u is made probability according to probabilities of available edges leaving u.

In this paper we are proposing an efficient algorithm for sound/speech synthesizer.

The algorithm uses speculatively parallelized techniques with multithreading and speculative computation reuse. We propose architecture of the model as the S3model represented as speculatively speech synthesizer.
The S4 model architecture is as follows.

The model works in three states. 1. Accepting state 2. Rejecting state 3. Dependent/Independent state.

We are considering the pattern consisting of the two sentences
I am A girl. and I am studying in school.

When two such sentences are spoken by a person in a language, if the speech matches with the pattern found the return value is 1 else the return value is 0.

Once the pattern is matched, that matched pattern is stored into the index table for the speculative reuse. When the person speaks again the same sentences the checking of the pattern and fetching them from the memory is avoided.

For speculative reuse computations value prediction table is referred which consists of
PatternID for each of the patterns.

PatternID	Pattern
P1	I am a girl
P2	I am going to school

All the PatternId's are represented through the index table which is organized as follows.

Index	PatternID
I1	P1
I2	P2

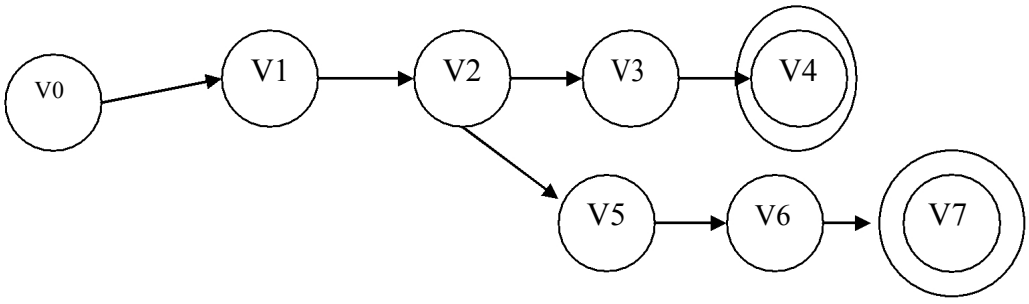

Fig.1 A graph showing the path of execution of the patterns mentioned above.
V0- Start vertex, V4,V7- Accepting state V0,V1,V2,V5 –Dependent state.

During the compilation, the compiler refers to value prediction table to get the PatternID whose match is present in the table. if the PatternId is retrieved, return value is 1 else it is 0. Once the patternId is retrieved, it's index is referred in the index table, based on the index value the compiler recognizes the matching patterns. i.e., the computations are reused without the compilations saving in time for the execution of the patterns for the matching.
The speculative multithreaded execution is as follows

The non speculative thread starts executing the tasks when it has to execute multiple instructions; it spawns a new thread and computes the speculative reusable instructions In the thread units which are speculative in nature. If the return value from the thread is 1 then the nonspeculative thread executes the next instruction considering the output from the speculative threads. If the return value is 0, the thread is squashed and nonspeculative thread starts executing the instructions normally.

```
Algorithm speechsynthesis (G(V,E))
 Input: A directed graph with n vertices and edges
where each edge represents the speech/sound sequence.
Output: The path of matched patterns.

For (i=1 to n)
For (j=1 to n)
Scanf (A[i,j])
The adjacency matrix is generated
 Call the random number generator.
Random walk ()
A non speculative thread starts execution
Visit [1]=1
P[x] array holds patterns
For (i=1 to n) do
It spawns speculative threads.
  If(visit[i]==1 && p[x]==b(y))
    X=x+1
Return 1
State pattern is found and matched
Else
Return 0

State pattern is not matched start normal execution
of the instructions. and squash the thread.
```

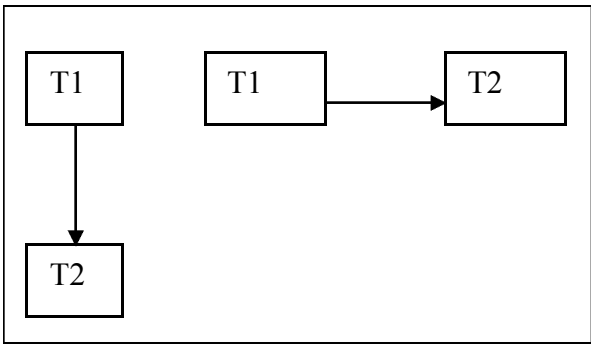

Fig.2 Normal Execution &Speculative parallel and computation reuse T1,T2-Tasks.

Analysis

The sequential computation time is T(n)=n units But parallel computation time is T(n)=T(n/2)+C(n)
C(n)=communication time. C(n)=1

As we use speculative reuse computation, there is no execution time in the speculative threads. Hence the execution time still reduces.

III Thread interval for Reliability

On the probability of occurrence of the digraph, we determine the future response of the graph by probability theory as the thread interval. An interval predictor predicts the interval at spawning interval or the loop interval.

Spawning interval is point of thread start point and speculation point. A process which has any interval arises for the future value of observation. i.e., a value speculation.[1,2,5,6,9]

Predicting the future value to be moved from producer thread to the consumer thread as it should be the accurate observation, based on spawning interval. One can predict the future value as well as assign confidence to the estimation so that the value predicted is more accurate. The thread interval can be predicted based on the partitioning samples. And the number of threads spawned. Considering the profiled information of partitioned samples, we calculate the mean and variance. We represent the mean as M and variance as V^2 the new estimator for the future value predictor is $P= V^2/PS$ here PS represents the Partitioned Samples.

Spawn Interval= (spawning point- thread start point)
Si= ts-tsp/ V^2+ V^2 /n .
Si =ts-tsp/1+1/n and Si is ranging from 0 to 1.
The probability of occurrence of the interval is determined as 1-b.
So the probability of value predicted for the speculation for future use is
tsp-v1+1/n < ts < tsp+v1+1/n

The total mean and variance from the normal distribution is considered. The mean shows number of spawns of the thread execution and variance is the V^2 verification of the introduced interval is such that a random walk is carried out on the digraph. We are determining the interval time at which the value lies between the accuracy of the prediction of the future value is accurate. Suppose ts=0.02 ms tsp=0.01ms n=10 spawns V=0.001. [3]
0.01-0.0011+1/10
96.72<103.48

Weibull distribution is basically applied for the reliability testing and for estimating the survival or life time of the thread. In this problem we are checking for the estimation of the reliability of the spawned thread without misspeculation. Because misspeculation causes the execution of the system to fail and again the normal execution of all computations should happen. To apply weibull distribution to estimate the rate of failure the probability that it will function properly for atleast a specified time under specified experimental conditions.
If s(t) is defined as the reliability of the spawned thread then $s(t)=p(T>t)=\int_t^\infty f(t)\,dt=1-f(t)$ where f(t) is cumulative distribution of T. [3]
The condition that the thread fails in the interval T=t to T=t+Δt, so that the thread has survived for the time t is

F(t+Δt)-F(t)/R(t)
Considering limit as Δt->0 we get the rate of failure denoted as $S(t)=\lim_{\Delta t\to 0} F(t+\Delta t)-F(t)/\Delta t\ 1/R(t)=f(t)/1-F(t)$

which is failure rate. Assuming that R(t)=1-F(t) and R'(t)=-F'(t)
we define the failure rate as s(t)=-R(t)/R(t)=-d[ln R(t)]/dt
$R(t)= c\, e^{-\int s(t)dt}$

The Weibull distribution with alpha and beta is defined as

$$f(x)=\begin{cases} \alpha\beta x^{\beta-1} e^{-\alpha x^{\beta}} & x>0 \\ 0 \quad \text{otherwise} & \end{cases}$$ where $\alpha > 0\ \beta > 0$ [5]

IV CONCLUSIONS

The lifetime or survival time of the threads are determined and we are proposing a solution to the open research area of determing the life time of the thread that the value should move from the producer thread to the consumer thread.

REFERENCES

1. Speculative execution in high performance computer architectures by David Kaeli and pen-chung yew. chapter 11,12,13 Chapman & Hall/CRC and information science series.

2. Principles of compiler design by Alfred V.Aho and Jeffrey D.Ullman. Chapter 12,13

3. Probability and Statistics for engineers and scientists, Seventh Edition, Ronald E.Walpole,Raymond H. Myers, Sharon L. Myers ,Keying Ye by PEARSON education Chapter 3&6.

4. P.Marcuello and A. Gonzalez,"control and data dependences speculation in multithreaded processors",
Proc.of the workshop on multithreaded execution, Architecture and compilation held in conjunction with HPCA-4, 1998.

5. J.Gonzalez and A. Gonzalez, "speculative Execution via Address Prediction and data prefetching", Proc of 11th ACM int.Conf. on Supercomputing,pp.196-203,1997.

6. J.Gonzalez and A. Gonzalez,"The potential of data value speculation to Boost ILP", Proc. Of Int.Conf. on Supercomputing,1998.

7 Y.Sazeides, S.vassiliadis and J.E.Smith,"The performance potential of Data Dependence speculation and collapsing" Proc.of the 29th int. Symp. On Microarchitectures,pp238-247,Dec.1997.

8. Y.Sazeides and J.E. Smith ,"The predictability of Data Values",Proc. Of the 30th .Int. Symp. On Microarchitectures, pp.238-247, Dec. 1997.

9. Y.Wu,D-Y.Chen, and J Fang. Better exploration of region-level value locally with integrated computation reuse and value prediction. In Proceedings 28th Annual

International Symposium on Computer
Architecture,pages98-108,jun.2001.