Using history and stride for TLB prefetching

Yash Patel, Arun Kanthali, Naman Verma, Ashish Raghuvanshi

(yashpatel, arunkanthali,namanverma,rashish)@iisc.ac.in

Gitlab Link: https://gitlab.com/yash42patel/tlb_prefetching.git

Summary

A large amount of execution time is spent in page table access as we have to access page table only in case of tlb miss , so if we reduce tlb miss we are going to improve execution time by a significant amount . So a lot work has been done on tlb's and prefetching is one of those areas but there is not much have been done in tlb prefetching as compared to cache prefetching so we are working on history based tlb prefetching which is not done and as from caches and branch prediction we get an intuition that history is repeated in that same order a large number of times.

So we have tried to analyze the same for SPEC 2006 benchmarks in gem5 and get results related to availability of patterns and how we can implement it feasibly by modifications.

1.Introduction

We have analysed address patterns and how our proposed scheme could be designed,so for that we analysed patterns based on history and got result which could be used in implementation and comparing with other prefetching techniques. We have analysed on last few bits and observed that how much false+ve or aliasing we are going to get but as there are design constraint so we have to choose tradeoff between wrong prefetching vs size of prediction table and register used.

So we have analysed on:

1.How much constant stride based patterns we are getting.

2.How much frequency of n consecutive accesses we get, means how much association of history is there in benchmarks.

Report is organized further as, In section 2. we have shown background and related work and how they have done .then we explained in section 3.that what are motivation behind our design and then in further remaining sections we have explained what we have done and what else can be done in future and what we are going to get out of it.

2. Background and Related Work

Address translation, done with virtualization and on larger application working sets is a major source of performance degradation. Many aspects of TLB, namely its structure, lookup etc have gone through rigorous studies but most of the sophisticated techniques to boost TLB performance use TLB prefetching, focuses on pre-fetching the next page in TLB before its actually called for on the basis of certain patterns or strides. This section gives brief overview of different TLB pre-fetching studies to provide deeper insight into TLB prefetching.

2.1. Recency based TLB pre-fetching:-[1] This technique exploits the temporal locality of TLB access such that a single LRU stack is maintained to store the temporal ordering of page accesses. This LRU stack, made up of doubly linked list, which stores a translation table entry consisting of two extra pointers, namely prev and next is used to predict the next pre-fetched entry.

2.2. Going the distance for TLB pre-fetching:-[2] an application driven study :- Distance prefetching works on the fact that markov and recency based prefetching takes more space for recording the pattern in reference behaviour and give a solution which does the same thing in a smaller space. It keeps tracks of the strides between successive references to make predictions which is helpful in case of spatial locality.

2.3. Inter core co-operative TLB pre-fetchers for chip multi-processors :- [3] Inter-core cooperative pre-fetchers are developed to exploit commonality in TLB miss patterns across multiple cores. This method can collectively improve 19% to 90% data TLB misses across the workloads.

2.4. Data cache prefetching using global history:-

[4]prefetching with global history buffer improves performance of cache and increases correlation prefetching performance by almost 20% and improves performance of stride prefetcher by 6%.

3. Motivation

Asspecifiedearlieralsotlbaremajorpartofmemorysystemofcpuasinclusionoftlb'ssave lotofpagetranslationtimeon which on average 40% of execution time is spent **[2].**So even a small improvement in tlb hit rate is going to improve performance of system a lot.

And second thing is why we are trying to do history based prefetching is as few techniques usinghistoryare implementedinthedatacacheslike**[4]**and haveprovided significantimprovementintlbandoverallperformance.Soweare tryingtousethishistorytype or a pattern which is likely to occur together every time.

4. Project Work :-

4.1. System specifications :

We have used gem5 simulator to get a simulation environment to run our workloads and to make changes in tlb configurationtoanalysebehaviorof tlbonvariouschanges.We workedonx86 64bitingem5. And our analysis was on **SPEC 2006benchmarksuite,**in SPEC 2006 suite we ran a few benchmarks like bzip , MCF , sjeng ,spec_rand 998 and 999. and other thanthosespecifiedabovewehaveranfewotherbuteithertheyarenotgivingenoughdatain significanttimeortheyseemed notcorrecttoussowehavedoneourwholeanalysisonthese 5.

4.2. Methodology :

Our approach is to record and analyse the patterns in the reference behavior.
Reference behavior can also be viewed broadly under one of these categories:

(a) showingregular/stridedaccessestoseveraldataitemsthataretouchedonlyonce

(b) showingregular/stridedaccessestoseveraldataitemsthataretouchedseveral times

(c) showing regular/strided accesses to several data items, but the stride itself can change over time for the same data item

Not having constant strided accesses (either keeps changing constantly or there is noregularityinthestrideatall),butrepeatingthesame irregularityfromoneaccessto another for the same data item over time.

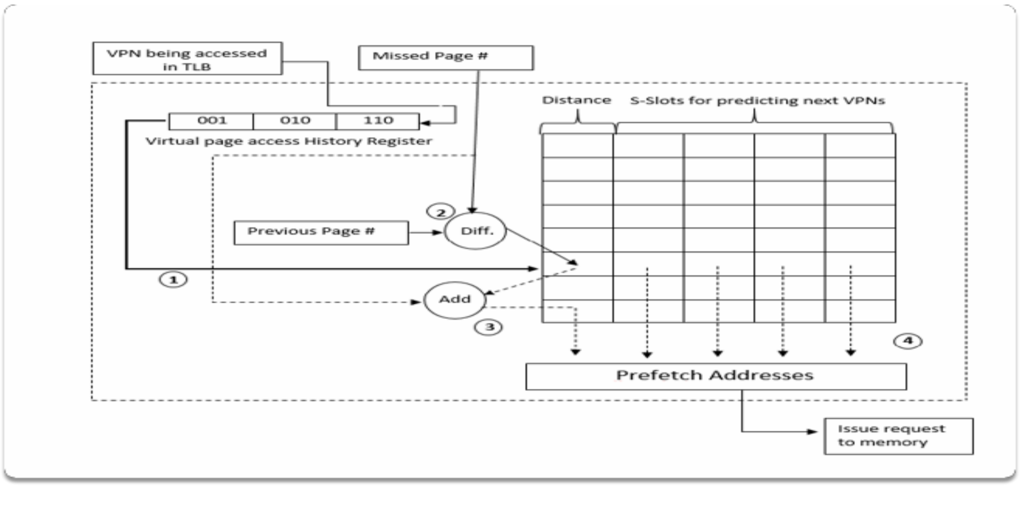


Figure: 4.1 Design of proposed TLB prefetch logic

In this design we exploit temporal locality along with the previous-stride to pre-fetch TLB entries. For this, a pre-fetch logic (Figure. 4.1) is designed consisting of a 'virtual page access history register' in which the history of previous k-addresses, accessed in TLB is stored (as few bits of the virtual page number (VPN)). This register indexes the entry of the prediction table. Prediction table contains two parts, first one takes the distance between the missed VPN in TLB and the previously accessed VPN (first of the s+1 slots), whereas the other part contains 's' slots with the VPNs that will be prefetched. Pre-fetching would include, the page numbers from these s-slots and one VPN that is calculated by adding the stride (stored in first slot) with the missed page number. Thus, this technique makes use of both the temporal and spatial locality of the TLB references for better TLB prefetching. In addition to it the Page Access History Register is updated at every new TLB reference by working analogously to a shift register with new entry being stored and oldest entry being discarded. In this we keep four prediction to be prefetched on a miss and they are going to be kept in buffer based on frequency of occurrence means if that particular sequence occurs frequently or not otherwise it's going to affect frequency of prediction a lot.

4.3 Approach :

Above specified is overview of how our implementation is going to work. for that we need to do profiling of patterns observed on various benchmarks

On basis of stride and history.

We have plotted results for various benchmark bzip , mcf , sjeng, spec_rand_998 ,spec_rand_99

4.4 Experimental procedure:

- **for patterns based on strides :** we are calculating total number of constant stride based pattern of different strides like length of stride and what stride is like 0,1,2,3. We have done all of this for 10 crore instructions in every benchmark if there are >=10 crore among our selected benchmarks. As this is simple technique so we are trying to analyse that could it be useful if used along with history in our design.

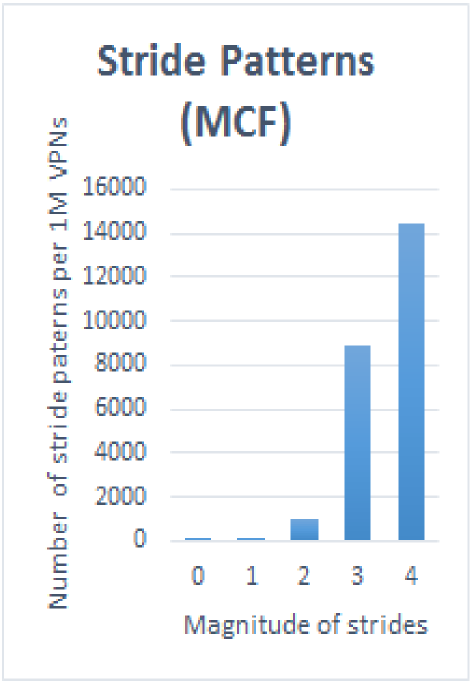
- **for patterns based on history of page access :-** to find repetition of history or say same set of VPN in particular sequence we are trying to find repetition of history locally, means in set of 20K instruction and we are analysing for 1 million tlb access as algorithm used by us has a n^2 complexity and it takes lot of time for even this 1million instructions.

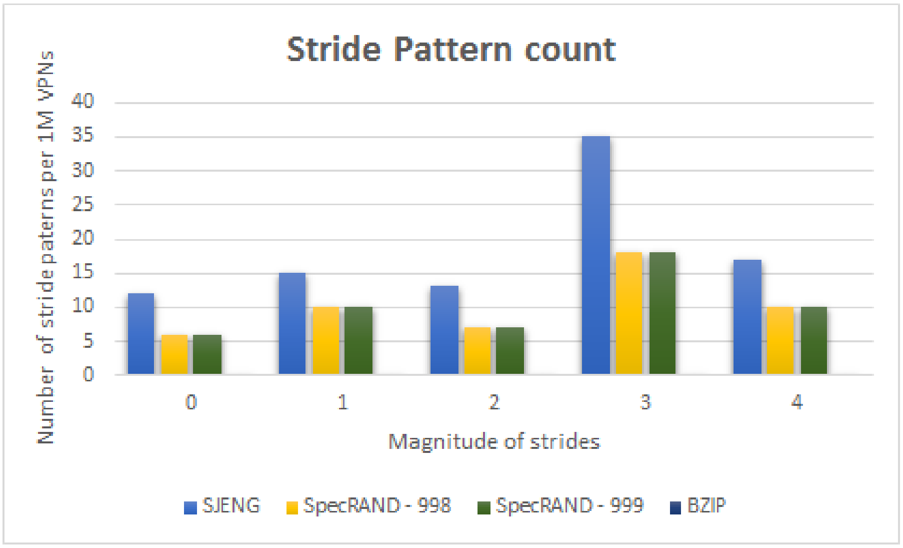
  Initially we are calculating actual repetition of sequence of VPN means using full page numbers but this is not feasible in designing as prediction table would be going to be very large than then again issue of size and latency of access of prediction table is going to affect performance so be are using few last bits of VPN to predict and prefetch. Because of this aliasing is going to be introduced so we select a tradeoff between size and false prediction on how many bits to be used for prediction so we have done how much wrong prediction increases as less no of bits are used.

4.5 Experimental results and evaluation :

In case of stride based patterns we donot have significant +ve results as we are using simple constant stride based prefetcher so that might be the reason. In this case we are only getting a lot of constant strides in MCF about 14000 strides of only length 4 so that's significant but in rest of benchmark we have got very few strides means in 10 crores of access we have got only 10's of strides. So this seems like it would not be going to add anything to performance when used along with history based prefetching .

.

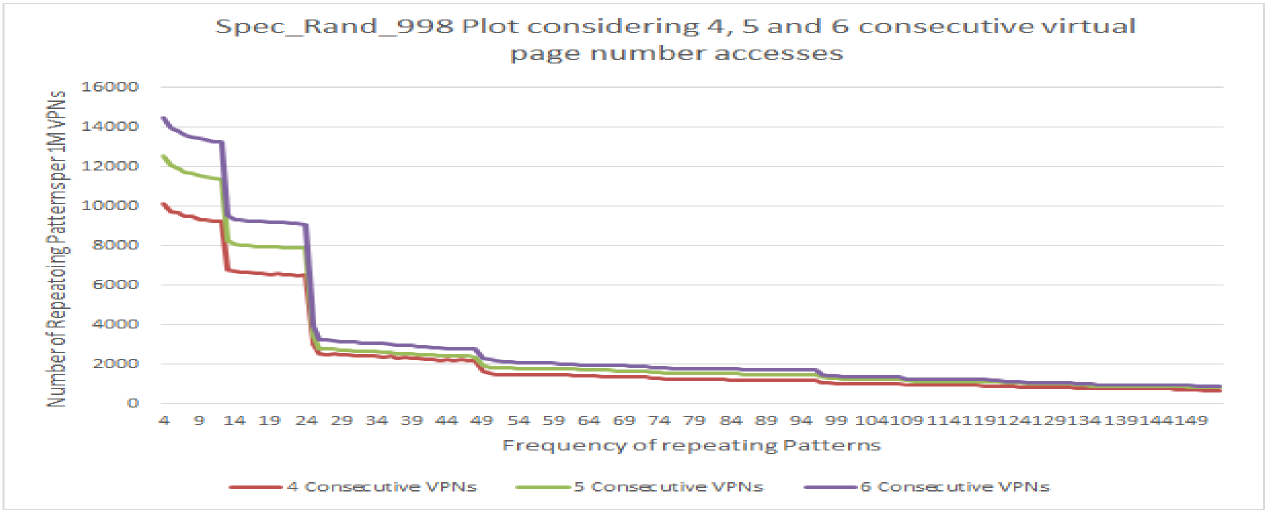


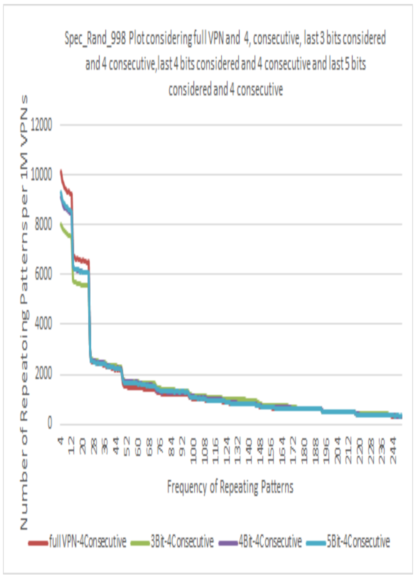


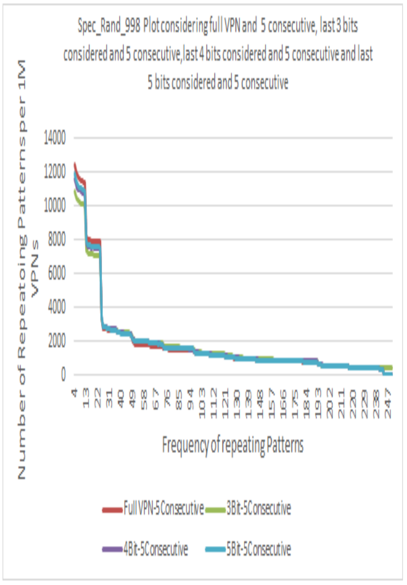
Graph for stride based analysis

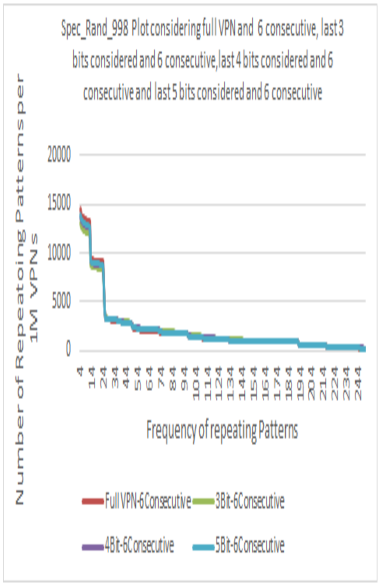
OBSERVATIONS:-
Incaseof historybasedpatternanalysis weareobservingthatavery highnoofwrongprefetcheswewould havetomakeifweareusinglastthree bitsforpredictionandprefetchingsoit'ssomethingwejustdoesnot want because this is going to remove actual history based entry from buffer and going to do lot of wrong history prediction that's not there. So we are also analyzing for last 4 and 5 bits that is what we are doing for checking history of length 4,5,6 asifweincreaselengthofhistorywearegettingassurancethattheyare more likely to occur in same order in future also , as from data we are getting that for example spec_rand 998 or 999 both shows better history frequency as compared to mcf .Same as in sjeng also we have not got that much total difference in higher frequency as compared to other but still means there is more no of sequence with higher frequency means in range of 100-250 in sjeng as compared to others.

1.Spec_rand_998

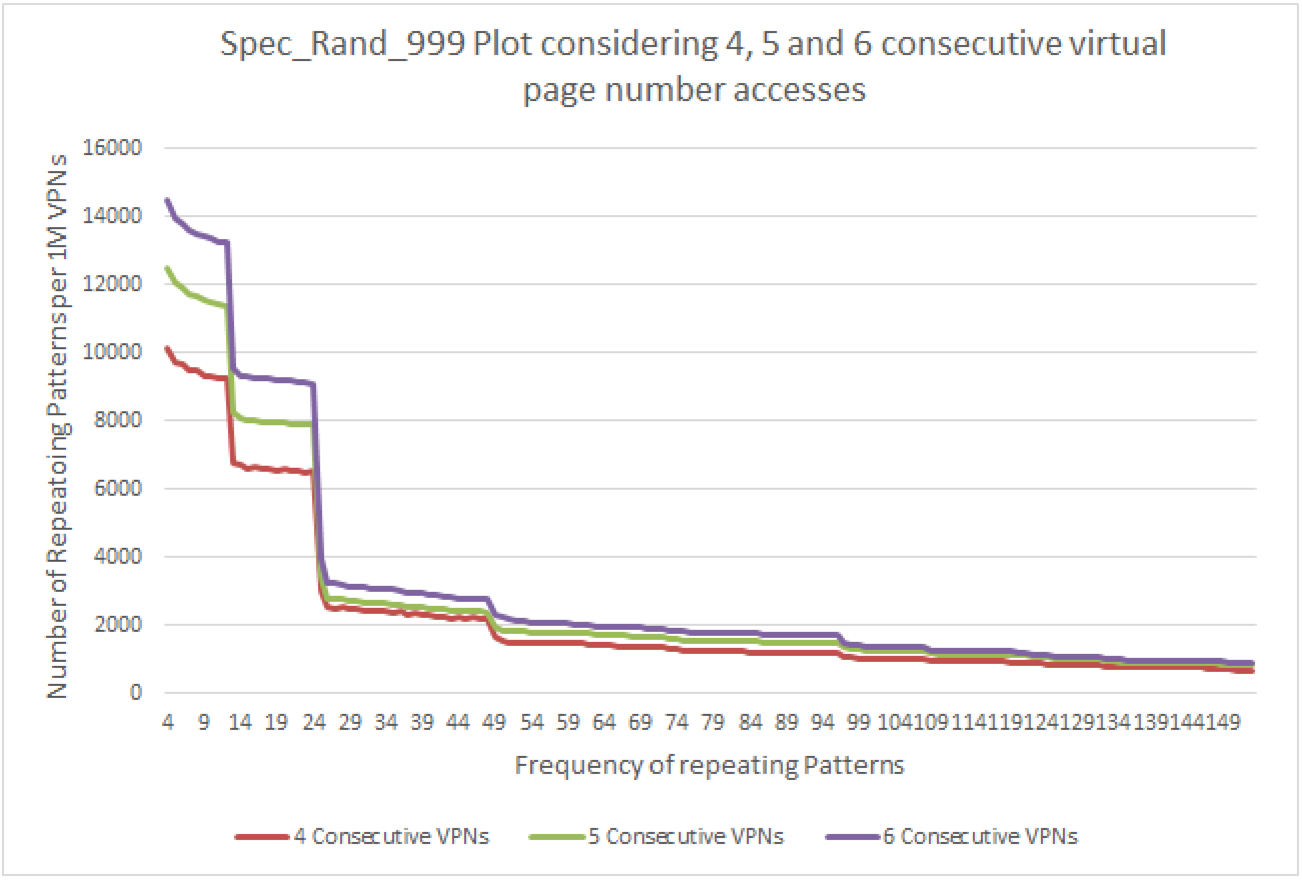


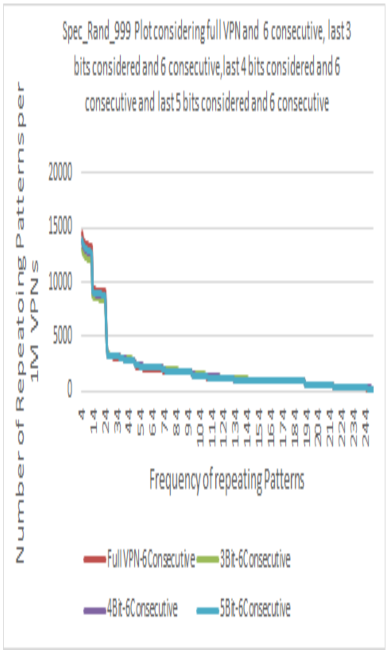
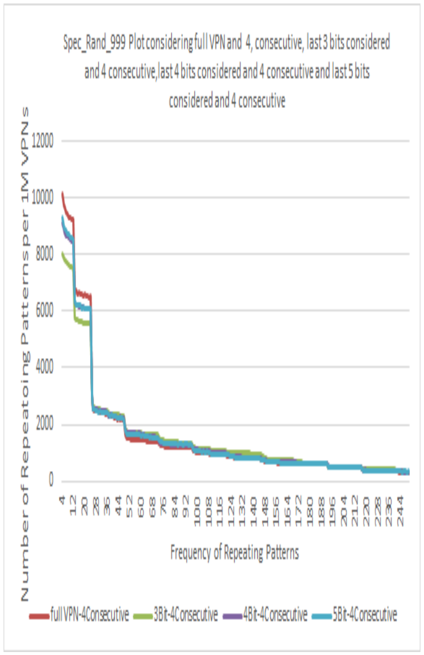
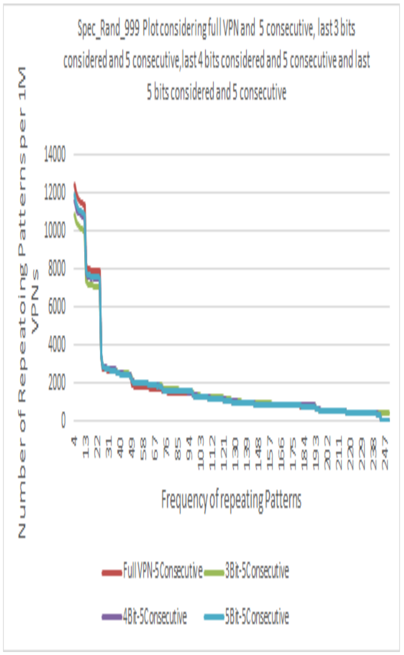




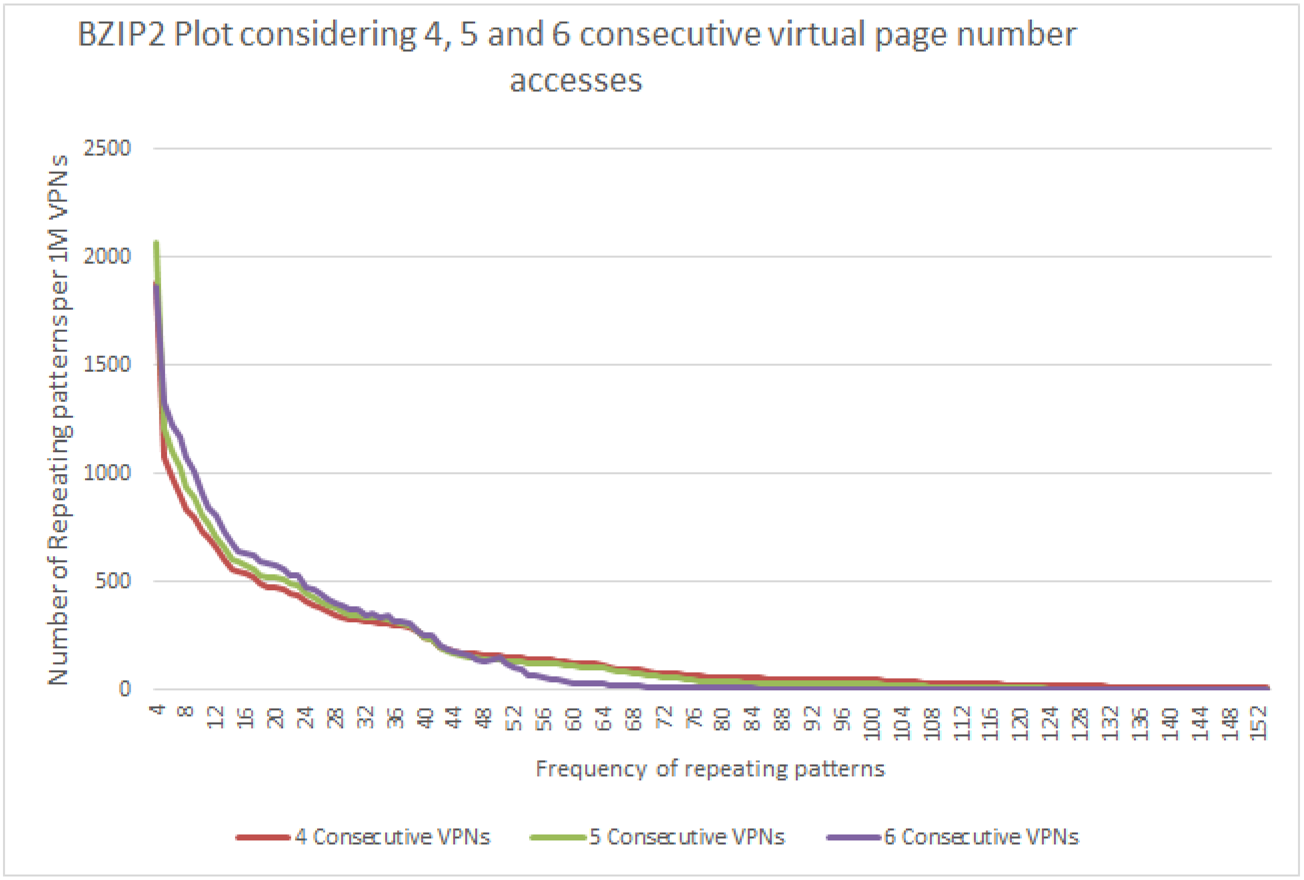


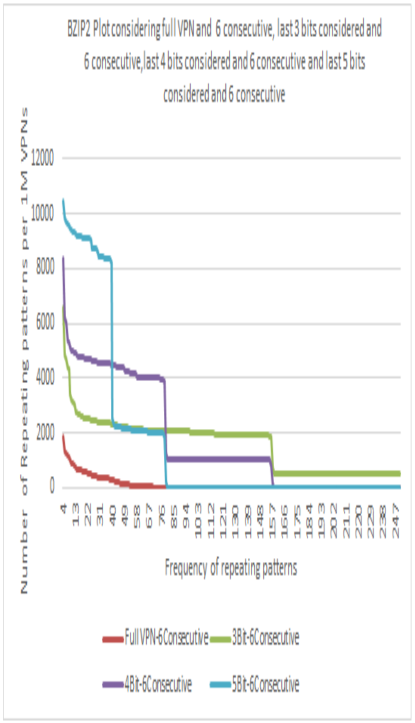
2.Spec_rand_999

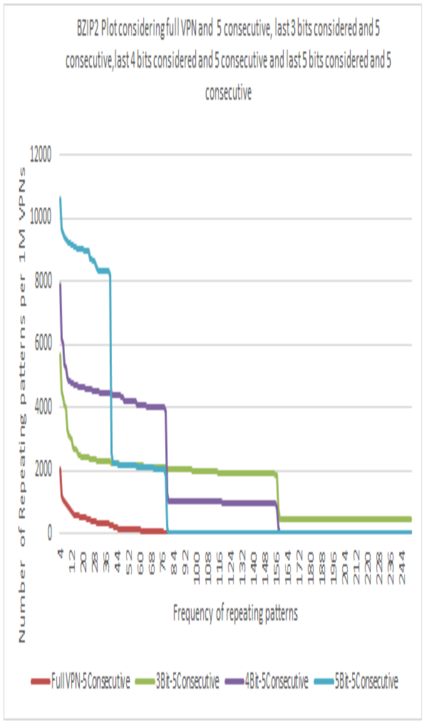


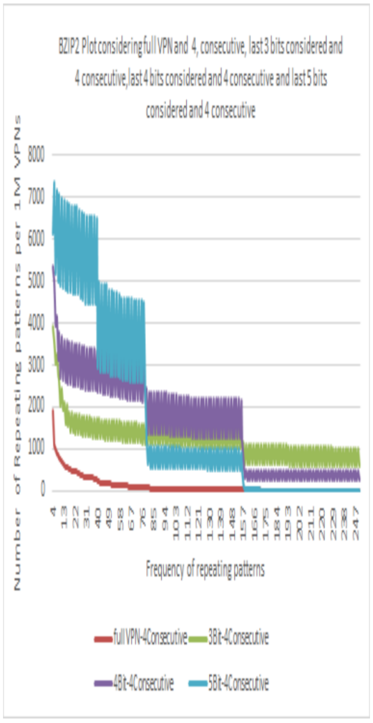

Spec_Rand_999 Plot considering full VPN and 6 consecutive, last 3 bits considered and 6 consecutive, last 4 bits considered and 6 consecutive and last 5 bits considered and 6 consecutive
Number of Repeatoing Patterns per 1M VPNs
Frequency of repeating Patterns
Full VPN-6Consecutive
3Bit-6Consecutive
4Bit-6Consecutive
5Bit-6Consecutive

Spec_Rand_999 Plot considering full VPN and 4, consecutive, last 3 bits considered and 4 consecutive, last 4 bits considered and 4 consecutive and last 5 bits considered and 4 consecutive
Number of Repeatoing Patterns per 1M VPNs
Frequency of Repeating Patterns
full VPN-4Consecutive
3Bit-4Consecutive
4Bit-4Consecutive
5Bit-4Consecutive

Spec_Rand_999 Plot considering full VPN and 5 consecutive, last 3 bits considered and 5 consecutive, last 4 bits considered and 5 consecutive and last 5 bits considered and 5 consecutive
Number of Repeatoing Patterns per 1M VPNs
Frequency of repeating Patterns
Full VPN-5Consecutive
3Bit-5Consecutive
4Bit-5Consecutive
5Bit-5Consecutive

3.Bzip2

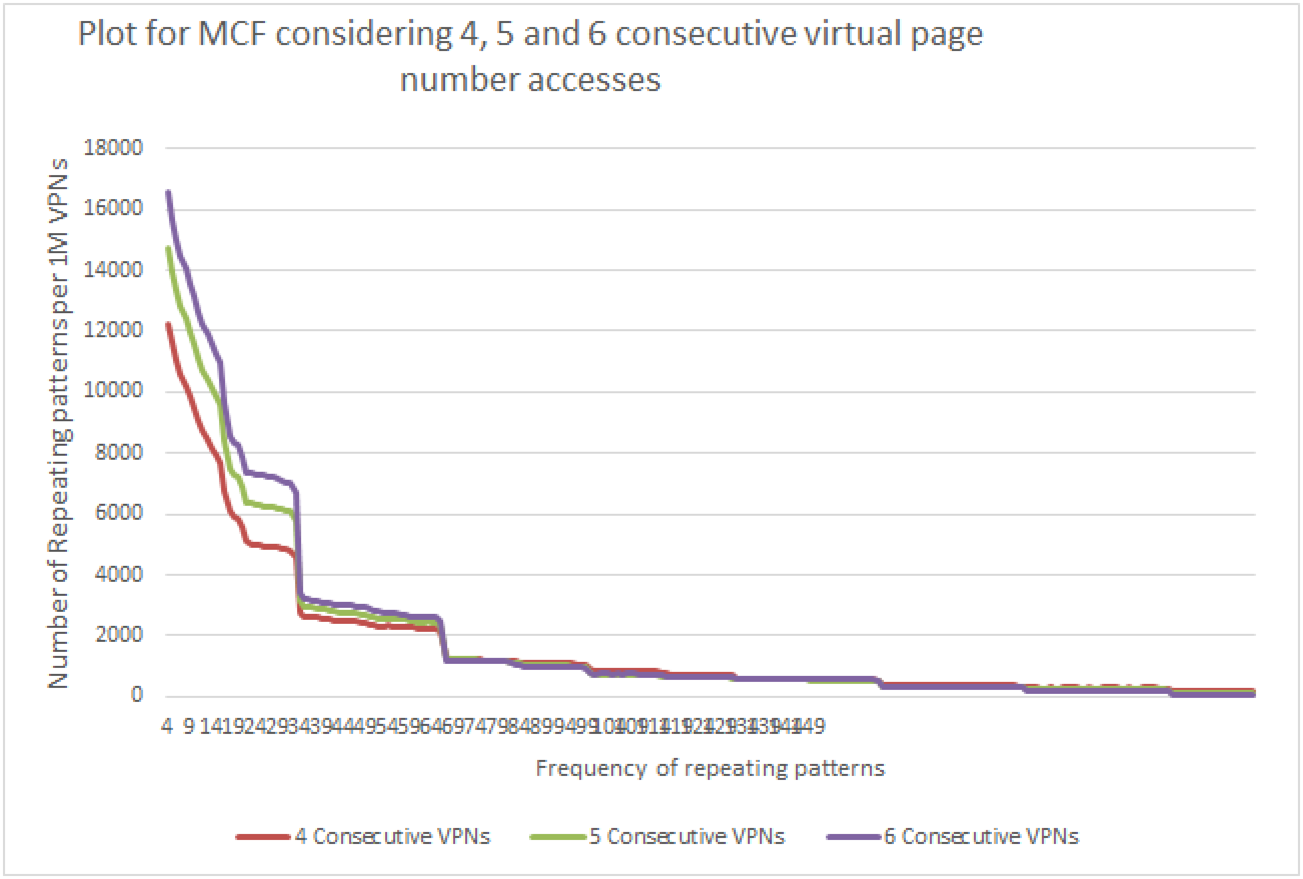

BZIP2 Plot considering 4, 5 and 6 consecutive virtual page number accesses
Number of Repeating patterns per 1M VPNs
Frequency of repeating patterns
4 Consecutive VPNs
5 Consecutive VPNs
6 Consecutive VPNs

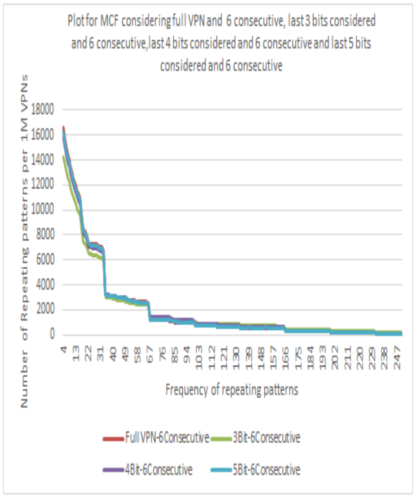

BZIP2 Plot considering full VPN and 6 consecutive, last 3 bits considered and 6 consecutive, last 4 bits considered and 6 consecutive and last 5 bits considered and 6 consecutive
Number of Repeating patterns per 1M VPNs
Frequency of repeating patterns
Full VPN-6Consecutive
3Bit-6Consecutive
4Bit-6Consecutive
5Bit-6Consecutive

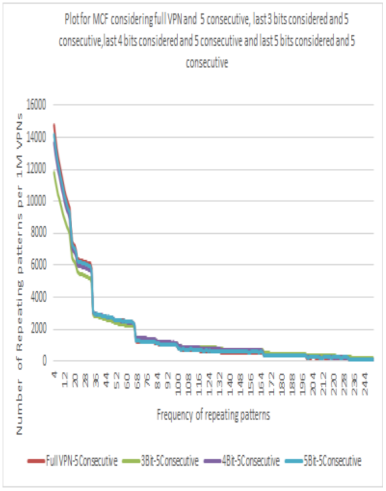

BZIP2 Plot considering full VPN and 5 consecutive, last 3 bits considered and 5 consecutive, last 4 bits considered and 5 consecutive and last 5 bits considered and 5 consecutive
Number of Repeating patterns per 1M VPNs
Frequency of repeating patterns
Full VPN-5Consecutive
3Bit-5Consecutive
4Bit-5Consecutive
5Bit-5Consecutive

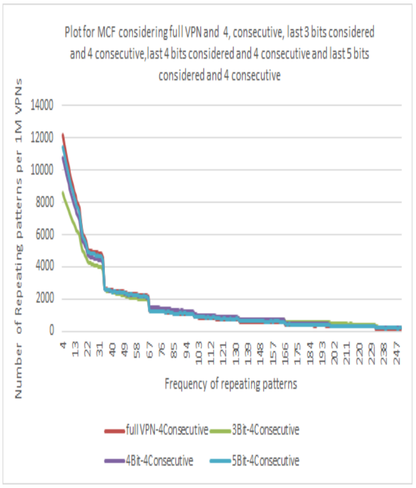

BZIP2 Plot considering full VPN and 4, consecutive, last 3 bits considered and 4 consecutive, last 4 bits considered and 4 consecutive and last 5 bits considered and 4 consecutive
Number of Repeating patterns per 1M VPNs
Frequency of repeating patterns
full VPN-4Consecutive
3Bit-4Consecutive
4Bit-4Consecutive
5Bit-4Consecutive

4.Mcf

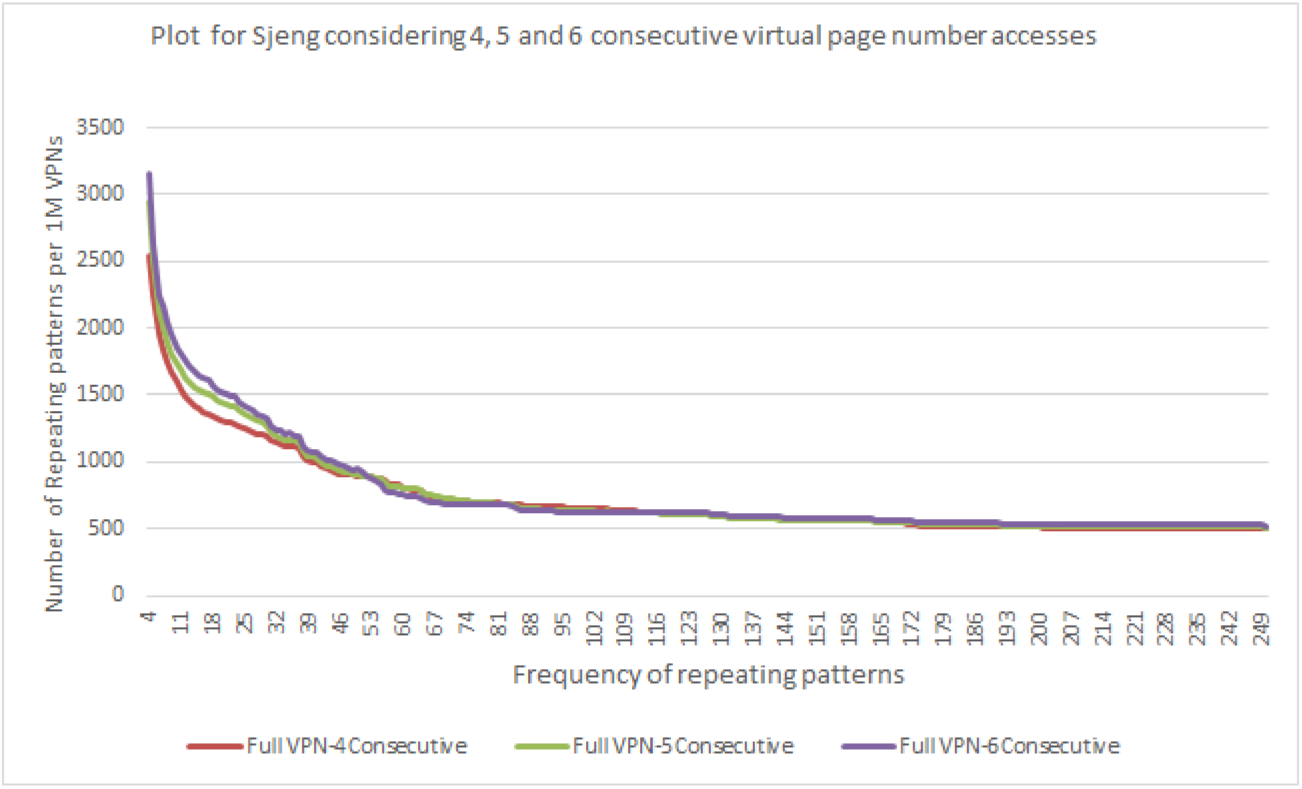

Plot for MCF considering 4, 5 and 6 consecutive virtual page number accesses
Number of Repeating patterns per 1M VPNs
Frequency of repeating patterns
4 Consecutive VPNs
5 Consecutive VPNs
6 Consecutive VPNs

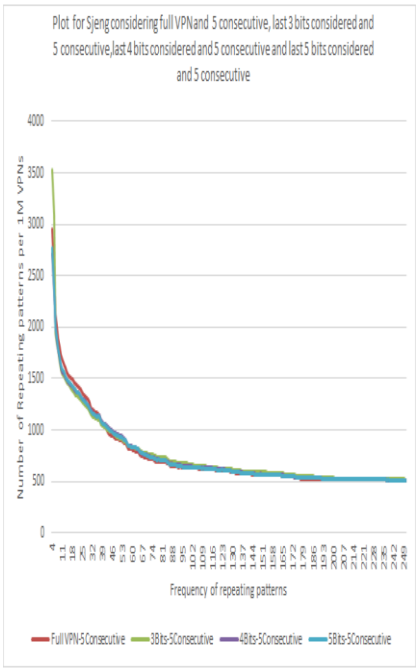

Plot for MCF considering full VPN and 6 consecutive, last 3 bits considered and 6 consecutive,last 4 bits considered and 6 consecutive and last 5 bits considered and 6 consecutive
Number of Repeating patterns per 1M VPNs
Frequency of repeating patterns
Full VPN-6Consecutive
3Bit-6Consecutive
4Bit-6Consecutive
5Bit-6Consecutive

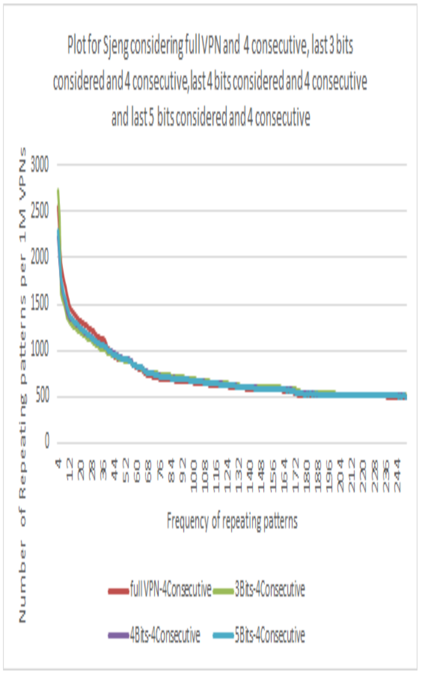

Plot for MCF considering full VPN and 5 consecutive, last 3 bits considered and 5 consecutive,last 4 bits considered and 5 consecutive and last 5 bits considered and 5 consecutive
Number of Repeating patterns per 1M VPNs
Frequency of repeating patterns
Full VPN-5Consecutive
3Bit-5Consecutive
4Bit-5Consecutive
5Bit-5Consecutive

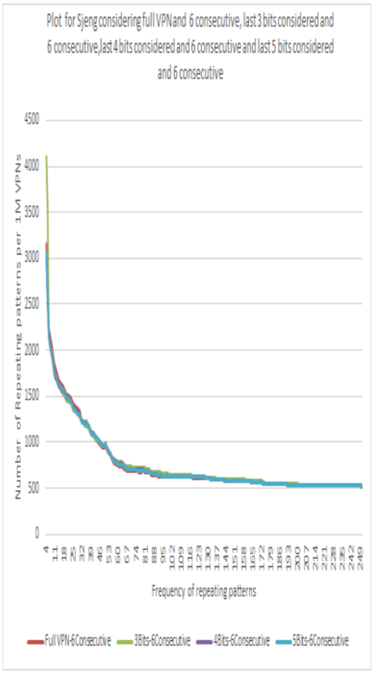

Plot for MCF considering full VPN and 4, consecutive, last 3 bits considered and 4 consecutive,last 4 bits considered and 4 consecutive and last 5 bits considered and 4 consecutive
Number of Repeating patterns per 1M VPNs
Frequency of repeating patterns
full VPN-4Consecutive
3Bit-4Consecutive
4Bit-4Consecutive
5Bit-4Consecutive

4.sjeng

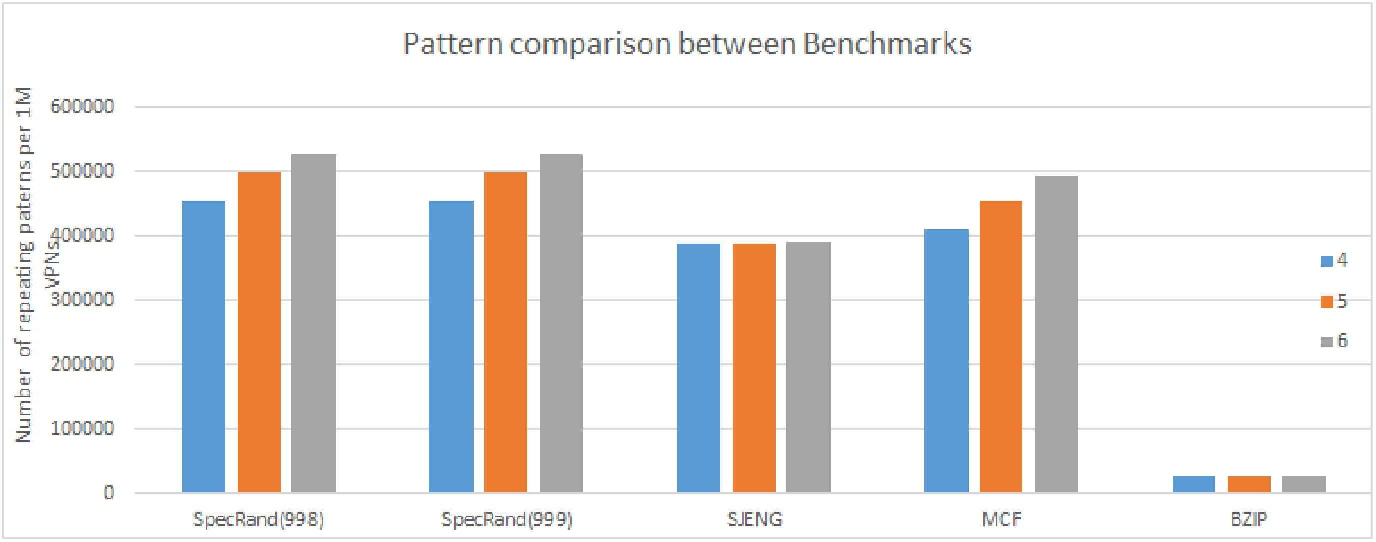

Plot for Sjeng considering 4, 5 and 6 consecutive virtual page number accesses
Number of Repeating patterns per 1M VPNs
Frequency of repeating patterns
Full VPN-4 Consecutive
Full VPN-5 Consecutive
Full VPN-6 Consecutive







Graph for comparison between various benchmarks



OBSERVATIONS :-

Here as we can see that bzip has very low amount of repetitive history whereas other 4 have good repetition but in them to sjeng has very high repetition of history as frequency is high in case of sjeng .Spec 998 and 999 have good amount of history repetition but frequency is not that high as compared to sjeng

4.6 Future work :-

As because of time constraints and lot of time getting spent on understanding gem5 ,it's working and setting up benchmark and amount of time simulatortakesforabenchmarkwearenot abletoimplementdesignbasedon our analyses ,so future work would we to implement scheme and see how it improves simple tlb performance or not ,so we have to implement given design on various parameters mentioned above for history based analyses and see how energy and cost increases and how much performance gain we are going to get we are getting any.

Than second would be to compare our schemes with other tlb prefetching techniques and how would our design going to work as compared to these prefetching techniques and how much misses our implementaion resolved which are not resolved in these implementations.

5. Acknowledgements

We would like to acknowledge Prof. Arkaprava Basu for his support and guidance for doing the project .We also like to thank few of our fellow colleagues and system lab members who have provided us lot of help in resolving few problems and We also like to acknowledge course e0-243 so that we were able to work on real problem.

6. References:-

[1] Ashley Saulsbury, fredrik.dahlgren,Per Stenström. Recency based tlb prefetching
[2] G.B. Kandiraju , A. Sivasubramaniam .Going the distance for TLB prefetching: an application- driven study.Proceedings 29th Annual International Symposium on Computer Architecture

[3] Abhishek Bhattacharjee and Margaret Martonosi .Inter-Core Cooperative TLB Prefetchers for Chip Multiprocessors .

[4] Kyles J Nesbit , James E Smith .Data cache prefetching using global history buffer.