Analysing Memory Address Streams Regularity using Signal Processing Methods

Samir AMMENOUCHE and Mounira BACHIR Sid-Ahmed-Ali TOUATI

University of Versailles Saint-Quentin en Yvelines, France

Abstract. The notion of data locality has been implicitely considered in high performance computing and code optimisation since many decades. The litterature presents multiple formal advances on static code analysis and optimisation for *regular* codes, while *irregular* codes are left for heuristics. Till now, the notion of *regularity* was not precisely defined, every approach of code analysis and optimisation may have its own point of view on it. This article presents a novel point of view on data regularity analysis using signal processing methods, allowing to analyse a larger fraction of programs that were considered irregular till now. We restrict our study on analysing the streams of accessed memory addresses during program execution, but the approach is general enough to be used for analysing other types of data streams in programs.

1 Introduction

Analysing the accessed memory addresses during program execution helps to improve and analyse the program performance. Different kinds of data can be collected during program execution: branch addresses, memory addresses, instructions addresses, data inside a memory location, etc. In this article, we consider the stream of data which is the accessed memory addresses during program execution. The characterisation of the accessed memory addresses is a well studied topic in the litterature that can be performed at compile time or at the run-time. At compile time, a static analysis of the source code can be performed to infer the accessed memory addresses. Indeed, the static code analysis at source level is until now limited to regular data access without any indirection (codes fitting in the polyhedral model). For the case of irregular memory access, static code analysis is most of the time ineffective. For this reason, dynamic on-line (during execution) or off-line (after execution) analysis can be a promising route. There are numerous mathematical tools used to precisely characterise and analyse data (accessed memory addresses), such as polynomial interpolation, different forms of regressions (linear, logarithmic ...). However, all these tools are focused on a restricted kind of programs. For instance, the linear regression fits with the collected addresses where the program issues a linear access to an array. A better tool is the polynomial interpolation which produces a function that resumes all the accessed memory addresses. However, the polynomial interpolation can potentially generate a set of polynomial coefficients equal to the highest term's degree. In other words, the number of polynomial coefficients which resumes the accessed memory addresses may be as large as the size of the data stream.
Regression models are also good candidates to perform accessed memory addresses analysis. In fact, a linear regression $y = a \times x + b$, for instance, can resume a large amount of data by few parameters such as: length of these data, the coefficients (a and b) of the affine function and the expected error. In addition, regression models give a pretty good performance on regular memory accessed addresses, and the noise around the main stream can be controlled. However, the linear regression is limited to accessed memory addresses which follows an affine function. In many other cases, the linear regression cannot be performed. This article proposes a new method which detects periodic regularities even in an *irregular* accessed memory addresses. In order to achieve the periodic regularity detection, three objectives arise:

1. Patterns detection: by resuming the input accessed addresses to a small number of parameters which contain the most significant part of the input.
2. Filtering noise: by looking for all the accessed memory addresses situated around a main stream but cannot be included in it. In other words, the noise is all the data which cannot be resumed by a regular pattern.
3. Compact the input memory stream: by detecting a regular patterns that involves a reduction of a large input by its pattern and hence reducing its size.

The aim of this study is to characterise the *best* memory stream. We focus only on a irregular accessed memory addresses and then perform a best effort analysis to achieve the criteria defined above. The results of this analysis can be used for multiple usages:

1. Accessed memory addresses analysis and prediction for software prefetching: If a pattern is clearly identified then we can easily use it to prefetch program's data from memory to cache.
2. Analysis the accessed memory addresses regularity: We define the notion of accessed memory addresses regularity in the next section. Generally, the regularity is associated with linearity.
3. Value prediction: if the accessed memory addresses analysis identifies precisely a pattern then its may be use for prediction.

In addition, the data analysis can be issued on-line or off-line and needs only a single pass. In order to study the efficiency of accessed memory addresses analysis, we generate several memory streams of linear algebra benchmarks focused on sparse matrix then we perform experiments on them. This article is organised as follows. Section 2 gives a precise definition of the notion of regularity, and gives tool to quantify this notion. Section 3 reminds the spectral analysis and its properties. Section 4 defines who and why using the spectral transform to perform the data analysis. Section 5 describes how the spectral analysis is used in the case of memory stream. Section 6 shows the obtained results using our the spectral analysis. And finally Section 8 concludes our work.

2 Background: Regularity versus Irregularity

In code optimisation field, the notion of data regularity is associated to linearity. If a linear increase or decrease of the accessed memory addresses is observed then the behavior of the load instruction is considered as regular. Furthermore, we find several definitions of the notion of regularity, depending on the research domain where it is applied.

2.1 Regularity in Information Theory

In information theory, the data regularity (data in broad sense) is described as the data entropy by Shannon [**?**] of Bell Labs. It is based on data occurrences analysis. The data entropy is defined by the following formula:

$$H(X) = -\sum_{n} P(x_i) \times log_2 P(x_i)$$

where: the different x_i are the sequence of the analysed data. In our field, the alphabet x_i is the memory stream addresses and $P(x_i)$ is the occurrence probability of the symbol x_i. In the information theory, the regularity approach is based on data redundancy, and it is usually used on the data coding, compression and transmission.

2.2 Regularity of Random Number Generator

In the field of statistic and probability, we need to generate different random numbers. In fact, an irregular sequence of numbers is defined and may satisfy a specified set of statistical tests [**?**][**?**].
The notion of regularity is defined through the notion of irregularity since the term irregular is the opposite of regular. By the way, statistical tests of random numbers generator are used to quantify the irregularity (and symmetrically the regularity). These tests quantify the degree of randomness on the analysed data sequence. In addition, from the different results we can symmetrically deduce the regularity of the sequence. In fact, the first test used for checking random number generation is the chi-square (χ^2) test. It can be issued for any kind of distribution. It is performed using the following formula:

$$D = \sum_{i=1}^{k} \frac{(o_i - e - i)^2}{e_i}$$

Where k is the number of generated random data, e_i and o_i are the expected and observed frequencies of the i^{th} data. The calculated D follows a chi-square distribution with $k-1$ degree of freedom. Using the chi-square table, the data can be classified as random or not.

A second statistical test which allows to assert if a sequence of numbers is random or not is the Kolmogorov-Smirnov test. This test defines two bounds

$$K^+ = \sqrt{n} \times \max(F_o(x) - F_e(x)) \text{where} x \in \mathbb{R}$$

and

$$K^- = \sqrt{n} \times \max(F_e(x) - F_o(x)) \text{where} x \in \mathbb{R}$$

Where: $F_e(x)$ is the expected cumulative distribution of the variable x and $F_o(x)$ is the observed one. n is the number of the observed variables. The K^+ measures the maximum deviation when the observed cumulative distribution function is above the expected one. While, the K^- measures the maximum deviation when the observed cumulative distribution function is bellow the expected one. The computed value of K^+ and K^- are used with the Kolmogorov-Smirnov table and permit to identify the α degree of randomness.

There are other statistical tests which permit to assert if the generated sequence of data is random or not. As observed, they quantify the degree of randomness of data compared to the expected one.

2.3 Granularity Effects in Regularity

In cache memory optimisation, an important criterion of analysis is the granularity of view (as camera zoom in/out in a picture). The regularity is a parameter which depends on the granularity. In accessed memory address analysis, the granularity is the manner to consider data, i.e the data (point) itself or a set of data. In the case of a set of data, the size of this set determines the level of granularity. We consider three levels of granularity of the accessed memory addresses:

1. The finest granularity is the address itself. For this level of granularity, we try to detect the regularity of the accessed memory addresses. In addition, we try to model the stream of memory addresses with a mathematical function or a statistical model if possible.
2. A cache-line granularity is the second level of granularity. At this level of granularity, the considered data becomes a cache-line and clusters all the addresses which are in the same cache-line to just one reference. The size of the cache-line depends on the target architecture. This case is interesting to develop because it indicates which cache-line to prefetch and the fact of clustering multiple references to just one cache-line hide the access order inside this cache-line. A simple way to get the cache-line granularity among a sequence of accessed memory addresses is to mask the low weight bits of the addresses and eliminate the consecutive duplicated addresses. Obviously, the collected sequence of the accessed memory addresses is viewed as temporal series. So hence, the order of this sequence is important to model.
3. A page or block granularity is the coarser granularity level we can consider. The size of the considered page of block must be smaller than the size of the cache memory. If the data are framed in a narrow window smaller than the cache then many optimisations are possible. Furthermore, the block must be atomic and cannot be divided into smaller blocks. As the medium granularity showed before, the sequence of the accessed memory addresses is a temporal serie where the order is an important parameter.

As said before, we call noise all the accessed memory addresses which cannot be resume by a regular pattern. The coarser grain granularity must not includes noise, i.e extra cache-line or block data outside the boundaries. If noise exists, the block must be divided into smaller blocks and the noise must be thought out. Furthermore, depending on the granularity level found in the sequence of the accessed memory addresses, three kinds of regularities can be detected.

1. Accessed memory addresses by linear sequence: this is the easiest kind of data to recognise. The input follows a linear/affine function. To proceed, all new accessed memory addresses are checked. The result is a set of linear/affine functions accompanied which their definition intervals. This set of linear/affine functions can be recursively clustered to construct an N dimensional polyhedron.

2. Another case of regularity which is possible to exploit is the fine irregularity in a bounded memory blocks. In this case, the accessed memory addresses seem fully irregular but still in a narrow window smaller than the cache. In opposition, these bounded memory blocks can follow a regular pattern. This can resumed by an irregular fine grain level granularity but a regular coarse grain level granularity.
3. If none of these methods is efficient and cannot respect the listed cases, a signal theory method can be applied to analyse the data. This method is called the spectral analysis and will be discussed in the next section.

3 Definition and Advantages of Spectral Analysis

The spectral analysis is a tool developed for the signal theory which allows to characterise an input signal by the frequencies [1] which compose it. From our point of view, it can be considered as a mathematical transformation, changing the sequence of the accessed memory addresses to the frequency domain.Among the advantages of spectral analysis, we quote:

- Detection of patterns: the spectral analysis permits to highlight the repeating patterns included in the analysed sequence if they exist.
- Characterisation of frequencies: It determines the granularity with which we observe the data, the high frequencies correspond to the fine granularities, while low frequencies correspond to the coarser granularities.
- It is a bijective function: the inverse transform exists, we recover exactly the same input signal if applying the inverse transform.
- Can be used in data compression: it allows to transform the input data to data with more entropy, which facilitates their compression (image compression example).

The spectral analysis can be issued by two ways:

1. The Discrete Fourier Transform (`DFT`) approach based on the Fourier Transform theory. It consists on a transformation applied to the accessed memory addresses and produces the frequency spectrum of these data. It transforms the original sequence of the memory accessed addresses to the corresponding sequence in the frequency domain. The mathematical definition of DFT is as follows[2]:

$$X_k = \sum_{n=0}^{N-1} x_k e^{\frac{-2i\pi kn}{N}}$$

   Where: x_0, x_1, ..., x_N are the sequence of the memory addresses accessed, The set of X_k are the obtained frequencies, N is the size of the input. e is the exponential function. k is the iterator on the accessed memory addresses. In practice, for N-point real input (memory addresses accessed are real value), it gives N-point complex output. The spectrum is the modulus [3] of complex number produced by the DFT.
   Furthermore, several fast Fourier transform libraries implement the `DFT` with a O($N \times log_2(N)$) complexity.
2. The Discrete Wavelet Transform (`DWT`) is similar to the Fourier transform. The Fourier transform decomposes the data to a sum of sinus and cosinus functions with different periods. In opposition, the wavelet decomposes the input of a sum of *mother wavelet signals* with different periods. The discrete wavelet transform is an O(N) algorithm and is known as the fast wavelet transform.

4 Data Analysis Using FFT/DWT

In this section, we describe the format of the input accessed memory addresses, Fourier transformation and Wavelet transformation, inverse Fourier transform and inverse wavelet transformation. In addition, we explain how to deal with noise and calculate the error rate.

[1] the cited frequency is different from CPU frequency, here frequency means the occurrences of repeating patterns
[2] The Euler formula link the sinus and cosinus functions to the exponential function
[3] the modulus is calculated by: $|z| = \sqrt{x^2 + y^2}$ where $z = x + yi$

4.1 Input Accessed Memory Addresses

The accessed memory addresses are collected following the following schema:
We issue a fine profiling focused on the cache misses instead of program execution times. This profiling is performed with the hardware counters. In addition, we identify the hottest code (in terms of cache misses). Furthermore, in the second time execution, we collect the accessed memory addresses using PIN for the Intel Core2 micro-architecture or the ST200run plug-in simulator for the ST231 processor. The targeted accessed addresses of the same load are clustered together. From this organisation, we obtain a set of loads, and for each one, a set of accessed memory addresses. In our study, we focus on the most delinquent loads. Let us define the delinquent loads by L. An instance $L(i)$ means that the load L is executed at the i^{th} time. A(L(i)) is the used address of the instance i of the load L. We represent the analysis of the accessed memory addresses by the set T_L:

$$T_L = A(L(i))|i = 1, n.$$

Where L has n instances. Once the input accessed memory addresses defined, we apply the Fourier transform to this input.

4.2 Fourier Transform

The collection of the accessed memory addresses is not necessarily periodic (***i.e.*** it is not exclusively composed of periodic patterns), thus the used period is the length of the input. The output of the Fourier transform is a vector of complex numbers. Each magnitude of the complex number corresponds to a frequency (the Fourier transform makes a transition from temporal values to frequencial one). Let us define Y_i the frequency obtained after the Fourier transform which is defined as:

$$Y_i = FFT_N(T_L)$$

where

$$FFT_N(T_L) = \{f_1, f_2, ..., f_N | f_k = \sum_{j=0}^{N} A(L(j)) e^{\frac{-2i\pi jk}{N}}\}$$

The input of the Fourier transform T_L (collection of accessed addresses) is real, the spectrum (in the frequency domain) is symmetrical. Therefore, for N real input, we analyse $\frac{N}{2}$ complex output. The other spectral analysis can be performed with the Wavelet transform which we describe in the following section.

4.3 Wavelet Transform

We use the Haar wavelet, as the Fourier transform, for accessed memory addresses characterisation and filtering. Figure 1 shows the mother wavelet of Haar wavelet. The mathematical definition of the Haar mother wavelet is:
$\psi(t) = \begin{cases} 1 & 0 \leq t \leq \frac{1}{2} \\ -1 & \frac{1}{2} \leq t \leq 1 \\ 0 & \text{Otherwise} \end{cases}$ The applied Haar transform function defined by:

$$\psi_{j,k}(A(L(i))) = \psi(2^j \times A(L(i)) - k); j \in \mathbb{N}; 0 \leq k \leq 2^j$$

Where j is the scale of the function, $A(L(i))$ is the accessed memory address and k is the shift. As the method described using the Fourier transform, the accessed memory addresses are decomposed with the Haar transform to a set of data of coefficients with smaller magnitude. The square shape of the Haar wavelet is more suitable to the integer data than continuous ones. This fact makes the Haar transform more precise and powerful than the Fourier transform. The noise filtering is the major advantage of the spectral analysis compared to the classical analysis (connectivity graph, markov chain. The spectral analysis associated with filtering is described in the following section.

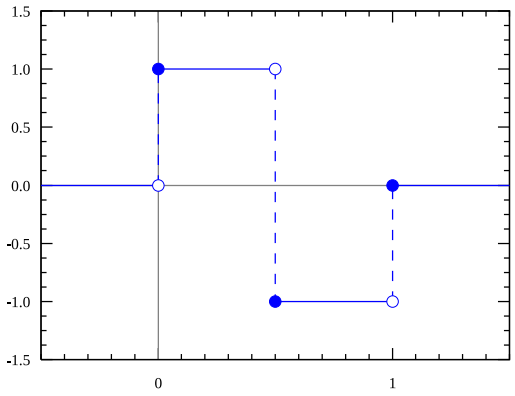

Fig. 1. The Haar Mother Wavelet ψ

5 How to Use the Spectral Analysis

The spectral analysis is based on the idea of repeating pattern of the strides deduced in the memory references. We apply the Fourier transform and the Haar wavelet transform according the following idea: decompose the input into a repeating pattern and separate it from the undesirable noise.

5.1 Fourier Transform Application

When applying the DFT, the input is transformed to the frequency domain. The analysis performed of the frequency domain has to handle the following characteristics:

- In the frequency domain, the output of the spectral transform shows a different aspect than the input. Figure2 shows the spectral transform of a random sequence repeated 1000 times. We can observe through this figure the frequency peaks. Indeed, narrow peaks on frequency domain implies a detected repetitive pattern and thus a large amount of the same sequence in the accessed memory address sequence.
- The peak amplitude has a direct relationship with the values of the data entries. The peak detection becomes problematic if the input data is composed of small values. The peak detection can be perturbed by noise which can has greater values. In an extreme case, the noise (not periodic pattern) can completely hide periodic input because of the amplitude values. In order to alleviate the problem, a possible solution is to associate weight to the periodical input values (which requires prior knowledge of the signal).
- For the analysis purpose, the highest peaks of the transform output are selected. They contain large panel of frequencies. The higher frequencies correspond to a short period pattern, for instance the most internal loop within a loop nest. In opposition, the lower frequencies correspond to a longer period pattern, for instance the outer loops.

The output of the spectral transform is splitted according to the peaks.In fact, the peaks represent the periodic pattern that we aim to analyse. The other values are considered as the noise which must be filtered. This approach is restrictive to near periodic pattern. We write Algorithm 1 that shows the peaks detection and noise elimination steps. For the peaks detection aim, the array `freq` containing the frequencies is sorted. This operation allows us to focus only on the `maxpeak` frequencies with the higher values. Furthermore, for each one of these frequencies with high values, the left and right frequencies which are around them are also selected. The left and right frequencies are selected in condition to be smaller than the central (with higher frequencies) with a `widthpeak` factor. The result is the `freq` array with only the higest peaks. In Figure 3, we show the used schema to analyse the accessed data memory references. At first, we apply a Fourier transform to the input data (strides sequence). Then we make a filter to the obtained output. This filter consists in keeping the N bigger values of the output. This is equivalent to select the frequency which contains the heaviest weight. The third part of this schema is optional. It consists on applying an inverse Fourier transform to

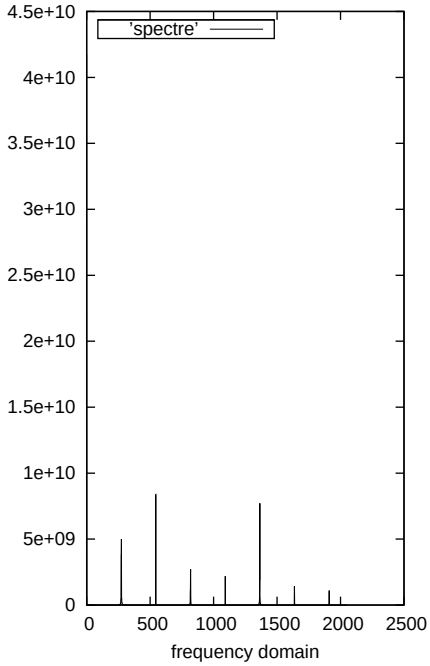

Fig. 2. Transform of the Random Sequence Repeated $\times$ 1000 times

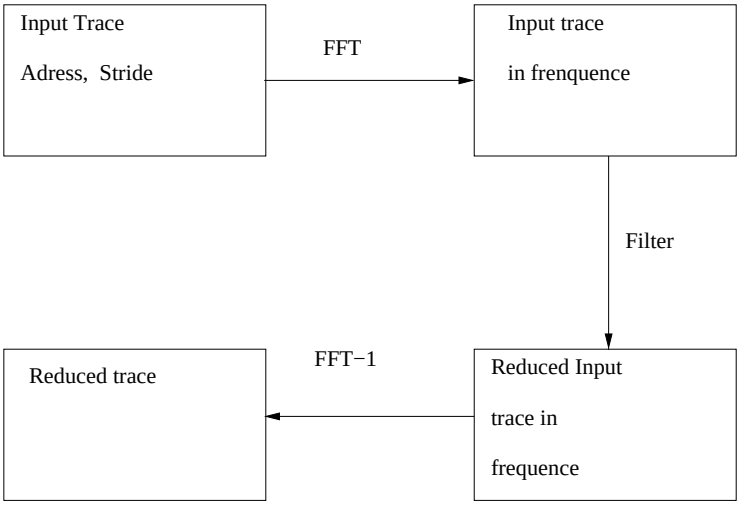

Fig. 3. The Accessed Memory Addresses Analysis Framework

Algorithm 1 Filtering Algorithm

```
Require: freq[N], widthpeak, maxpeak;
Ensure: freq[N];
  int T=sub_array(sort(freq),maxpeak);
  {T is a sorted index of freq}
  for (i=0;i < maxpeak; i++) do
    {loop to treat the highest peaks}
    j=0;
    while (freq[T[i]+j] ≤ widthpeak × freq[T[i]] do
      {loop to calculate the peak's width}
      WIDTH[i]++;
      j++;
    end while
    {WIDTH contains the width of the higest peaks}
  end for
  Nfreq[]=NewArray(SetToZero());
  {Copy only the highest peaks and its neighbor}
  for (i=0;i ¡ N; i++ do
    for (j= -WIDTH[i] ; j ≤ WIDTH[i] ; j++ ) do
      Nfreq[]=freq[i+j];
    end for
  end for
  freq[]=Nfreq[];
```

compare the obtained output with the input and then to check the quality of the analysis. One important parameter of the data analysis is the size of the frequency kept. It is, until now, a constant parameter, which is set by the user. However, it can be dynamically set by the framework.

5.2 Haar Wavelet Transform Application

The Wavelet transform is used in the same manner as the Fourier transform. The used approach is based on the Haar wavelet transform property: any real data input can be approached by a linear combination of the Haar scaling function. The number of those scaling functions determine the regularity/irregularity of the input. The heart of our idea is to apply the Haar transform to the sequence of the strides generated by the memory references. By the way, the obtained results are filtered as the same manner applied in the Fourier approach. The last step is optional for checking the efficiency of the filter. Let us see the quantification of the noise included in the input accessed memory addresses.

5.3 Error Rate

Our analysis method is based on filtering the obtained frequencies or Haar coefficients output array using respectively the Fourier transform or the Haar wavelet. One major parameter of the filtering is the loss ratio and the noise filtering. To measure the error rate, we must re-generate the input address sequence after filtering and compare the re-generated one with the original one. Depending on the set of collected addresses, the quality of filtering is induced. We need to care about the quality of the filtering, it is a trade-off between the loss rate and the filtering quality. The mathematical definition of the filtering quality is as follows.
The error rate is: $E = \frac{|\rho_L|}{|T_L|}$
with

$$\rho_L = \{j | A(L(j)) \neq A'(L(j))\}$$

Where ρ_L is the set of value that have a difference between the input accessed memory addresses $A(L(j))$ and the re-generated output $A'(L(j))$. This formula quantifies error rate when filtering the transformed input. It takes into account the order sequence and the values themselves (even also in the case of cache optimisation, if these values can

be clustered into cache lines). We can quantify the error by using the following formula:

$$QE = \frac{\sum_{i=0}^{|T_L|} |A(L(i)) - A'(L(i))|}{\sum_{i=0}^{|T_L|} A(L(i))}$$

It indicates how much the re-generated trace is different from the original one. It gives an idea of the importance of the quality filtering. QE gives finner information than E.

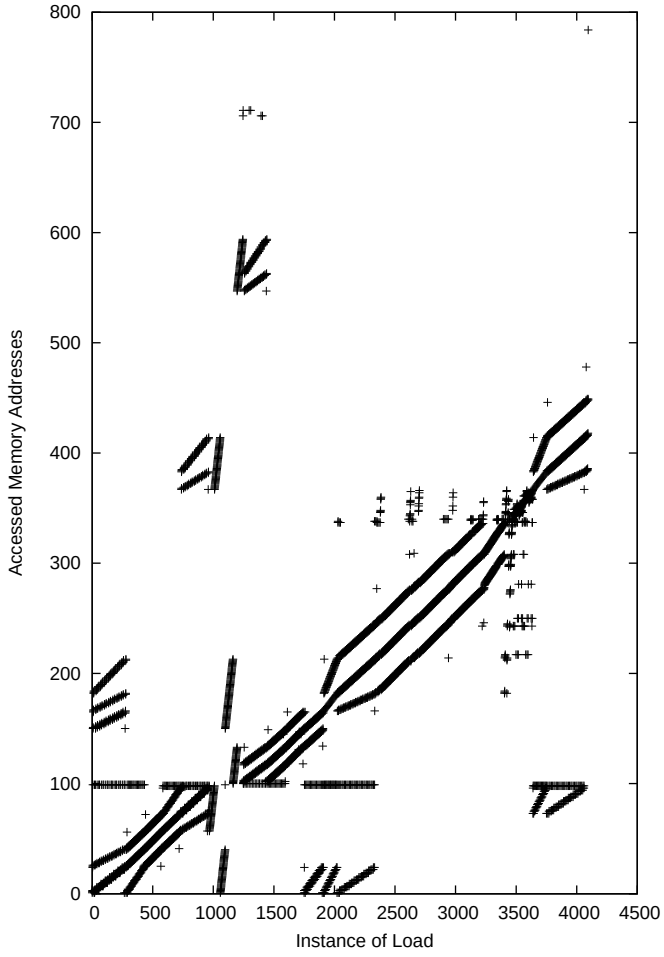

Fig. 4. Address Sequence of Sub-script Array of the Sparse Matrix Vector Multiplication

6 Experiments Results

Figure 4 shows the accessed memory addresses issued by the sparse matrix vector multiplication using the `Barth` sparse matrix available at [**?**]. This sparse matrix is used as a subscript array to access data in a sparse matrix vector

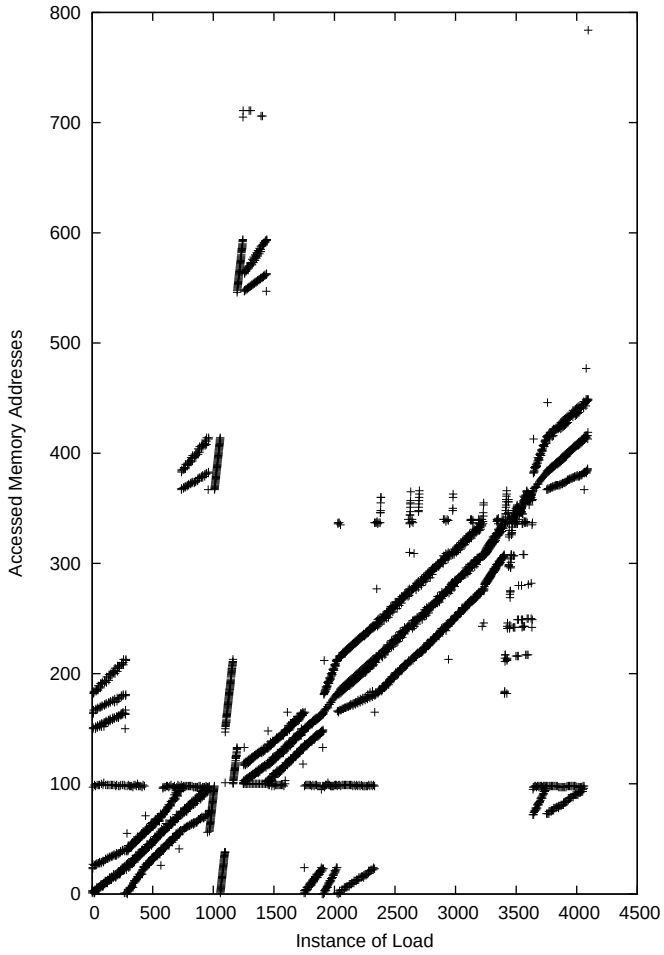

Fig. 5. Regenerated and Filtered Address Sequence of Sub-script Array of the Sparse Matrix Vector Multiplication Using FFT

multiplication. In addition, we perform tests on this matrix. The sequence of memory addresses accessed via this matrix is irregular. Figure 5 shows the results after applying our developed method: FFT transform combined filtering (described in Figure 3. A few repeating pattern are found, but at least the filtering operation does not degrade the data. In addition to the `Barth`, we perform experiments over a large set of matrices (1431 matrices) available at [**?**]. For the experiments, we consider four values for the $maxpeak$ length parameter, 10%, 25%, 50% and 75% of the total length of the analysed sequence of the accessed addresses. Figure 6 shows the boxplot of the error rate applying the

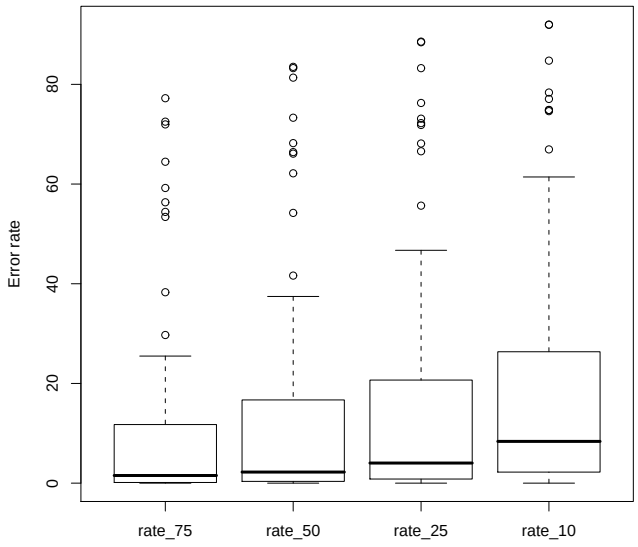

Fig. 6. FFT Error Rate Varying Maxpeak length from 75%, 50%, 25% to 10% of the Original Sequence length

FFT combined with filtering. Figure 6 uses 75%, 50%, 25% and 10% of the original sequence length as $maxpeak$ values. The X axis is the $maxpeak$ length (ratio of the original length) and the Y axis is the error rate (percentage). Figure 6 shows that in the worst case ($maxpeak$ is 10% of the sequence length), half of the experimented sequences have an error rate less than 10%. In the other case ($maxpeak$ is 75% of the sequence length), the third quarter of the experimented sequences have an error rate less than 20%. Figure 7 shows the boxplot of the error rate applying the Haar Wavelet combined with filtering. Figure 7 uses 75%, 50%, 25% and 10% of the original sequence length as as $maxpeak$ values. The X axis is the $maxpeak$ length (ratio of the original length) and the Y axis is the error rate (percentage). Figure 7 shows that in the worst case ($maxpeak$ is 10% of the sequence length), half of the experimented sequences have an error rate less than 5% which is a better result than the FFT result. Using 50% of the sequence length as $maxpeak$, the experiments show that half of the error rates are about 0. Finally, in the last case where $maxpeak$ is 75% of the sequence length, except few error rate values, all the other error rates are about 0%. Comparing Figure 6 and Figure 7, we can easily claim that the Wavelet results are better than the FFT one.

7 Related Work

The characterisation of the irregular data behavior (in general) and the irregular memory accessed addresses (in particular) is a well studied topic. In fact, the irregularity data can be described by the technique proposed by Shannon [**?**] called the data entropy which is based on data occurrences analysis. The data entropy is usually used for data coding,

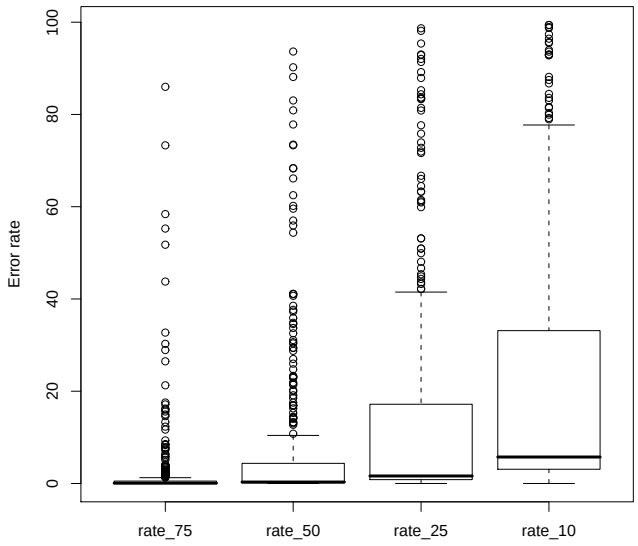

Fig. 7. Haar Wavelet Error Rate Varying Maxpeak length from 75%, 50%, 25% to 10% of the Original Sequence length

compression and transmission. In the statistical field, there are several tests (Chi-square, Kolmogorov-Smirnov,...) which determine if the generated data sequence is random or not. This definition can be symmetrically used in the case where the statistical tests fail to affirm if the data sequence is random or not. In this case, it may contain some regularities. Beyler *et al.* [**?**] use the Markov model to characterise the accessed memory addresses. Indeed, their first aim is to characterise the memory behavior by collecting the different information and then predicting the issued strides generated by the accessed memory addresses. Their major goal is to perform software prefetch using results obtained by the first aim. Furthermore, they propose an on-line portable and software solution based on a memory strides Markov model called *esodyp*. Markov model relies on a history of accessed data to predict the next data to be accessed. Beyler *et al.* perform experiments using ft, equake, art and mcf benchmarks of spec2000 and treeadd benchmark upon Itanium architecture combining both icc and gcc compilers. Ramamoorthy [**?**] proposes to use the connectivity matrix and the reachability matrix to store the accessed virtual pages or caches lines. Based upon these matrices, he proposes to perform the prediction of memory pages or cache-lines. Let us note that the Ramamoorthy's connectivity matrix has the same signification as in graph theory. In fact, we store all the connected nodes in this matrix. In addition, this matrix is used as the predictor of the next accessed page or cache-line. The reachability matrix stores for each node, all the reacheable possible nodes. This technique is used to mark the pages or cache-lines which must be replaced. The idea to save past sequences to predict the future misses is also used in the correlation prefetching. At first, this latter consists on storing several sequences on a table and then comparing the current accessed sequence with the stored one. This optimisation has been implemented by Baer *et al.* [**?**], Rechstschaffen *et al.* [**?**] and Charney *et al.* [**?**]. The loop trace recognition is performed by Clauss *et al.* in [**?**]. In fact, they start by performing a memory access profiling and then construct polytopes which are complex structures. These structures resume memory behavior, periodicity, linearly accessed linked memory and repetition.

8 Conclusion

In this article, we show that the data collection and analysis is a critical part of the prefetch process. Usually, the term "regular memory access" is used to define an access stream modeled by an affine function. We think that a repetitive

pattern can also be considered as regular. We propose a framework to detect repetitive patterns through irregular access stream. It is a software solution based on spectral analysis. We used the FFT approach and the Wavelet one combined both with filtering. We think that the Wavelet approach is more suitable, because it has a lower complexity O(N) and it brings better results in term of error ratio. We perform experiments in an irregular matrix, and we show the efficiency of our approach. Our approach can also be used to compact traces with low loss. The spectral analysis allows also the distinguish between the high frequencies and the low ones: the high frequencies correspond to a small pattern, the low frequencies correspond to a bigger pattern. The used benchmark, the sparse matrix vector multiplication, is a good candidate to perform a non affine prefetch due to the indirection which is the topic of our current work.

References

1. E.Shannon, C.: A mathematical theory of communication. Bell System Technical Journal **27** (1948) 379–423, 623–656
2. Jain, R.: The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling. Wiley (1991)
3. Knuth, D.: The Art of Computer Programming. Addison-Wesley (1997)
4. Davis, T.: University of florida sparse matrix collection : sparse matrices from a wide range of applications (March 2010) `http://www.cise.ufl.edu/research/sparse/matrices/`.
5. BEYLER, J.C., CLAUSS, P.: Esodyp: An entirely software and dynamic data prefetcher based on a markov model. In: Proceedings of the 12th Workshop on Compilers for Parallel Computers, CPC 2006, University of A Coruna (january 2006) 118–132
6. Ramamoorthy, C.V.: The analytic design of a dynamic look ahead and program segmenting system for multiprogrammed computers. In: Proceedings of the 1966 21st national conference. ACM '66, New York, NY, USA, ACM (1966) 229–239
7. Baier, J.L., Sager, G.R.: Dynamic improvement of locality in virtual memory systems. IEEE Trans. Softw. Eng. **2** (January 1976) 54–62
8. Rechtschaffen, R.N.: Cache miss history table. Technical report (April 1983)
9. Charney, M.J.: Correlation-based hardware prefetching. PhD thesis, Ithaca, NY, USA (1995) UMI Order No. GAX95-42429.
10. Clauss, P., Kenmei, B.: Polyhedral modeling and analysis of memory access profiles. In: ASAP '06: Proceedings of the IEEE 17th International Conference on Application-specific Systems, Architectures and Processors, Washington, DC, USA, IEEE Computer Society (2006) 191–198