

Fast Burst Correlation of Financial Data

Michail Vlachos, Kun-Lung Wu, Shyh-Kwei Chen, and Philip S. Yu

IBM. T.J. Watson Research Center
19 Skyline Dr, Hawthorne, NY

Abstract. *We examine the problem of monitoring and identification of correlated burst patterns in multi-stream time series databases. Our methodology is comprised of two steps: a burst detection part, followed by a burst indexing step. The burst detection scheme imposes a variable threshold on the examined data and takes advantage of the skewed distribution that is typically encountered in many applications. The indexing step utilizes a memory-based interval index for effectively identifying the overlapping burst regions. While the focus of this work is on financial data, the proposed methods and data-structures can find applications for anomaly or novelty detection in telecommunications and network traffic, as well as in medical data. Finally, we manifest the real-time response of our burst indexing technique, and demonstrate the usefulness of the approach for correlating surprising volume trading events at the NY stock exchange.*

1 Introduction

“Panta rhei”, said Heraklitos; everything is ‘in flux’. The truth of this famous aphorism by the ancient Greek philosopher is so much more valid today. People need to make decisions about financial, personal or inter-personal matters based on the observations of various factoring parameters. Therefore, since everything is in constant flow, monitoring the volatility/variability of important measurements over time, becomes a critical determinant in any decision making process.

When dealing with time sequences, or time-series data, one important indicator of change is the presence of ‘burstiness’, which suggests that more events of importance are happening within the same time frame. Therefore, the identification of bursts can provide useful insights about an imminent change in the monitoring quantity, allowing the system analyst or individual to act upon a timely and informed decision.

Monitoring and modeling of burst behavior is significant in many areas; first and foremost, in *computer networks* it is generally recognized that network traffic can be bursty in various time-scales [9, 6]. Detection of bursts is therefore inherently important for identification of network bottlenecks or for *intrusion detection*, since an excessive amount of incoming packets may be a valid indication that a network system is under attack [13]. Additionally, for applications such as *fraud detection* it is very critical to efficiently recognize any anomalous activity (typically in the form of over-utilization of resources). For example, burst

detection techniques can be fruitfully utilized for spotting suspicious activities in large stock trading volumes [10] or for identification of fraudulent phone activity [12]. Finally, in *epidemiology and bio-terrorism*, scientists are interested in the early detection of a disease outbreak. This may be indicated by the discovery of a sudden increase in the number of illnesses or visits to the doctor within a certain geographic area [16, 17].

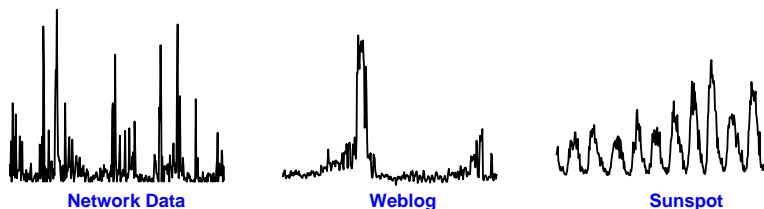


Fig. 1. Burst examples in time-series data

Many recent works address the problem of burst detection [19, 7]. However, in many disciplines, more effective knowledge discovery can be achieved by identifying *correlated* bursts when monitoring multiple data sources. From a data-mining perspective, this task is more exciting and challenging, since it involves the identification of burst ‘clusters’ and it can also aid the discovery of causal chains of burst events, which possibly occur across multiple data streams. Instances of the above problems can be encountered in many financial and stock market applications, e.g., for triggering fraud alarms. Finally, burst correlation can be applicable for the discovery and measurement of gene coexpression (in this field, burst appears under the term ‘up-regulation’), which holds substantial biological significance, since it can provide insight into functionally related groups of genes and proteins [5].

Addressing the above issues, this paper presents a complete framework for effective multi-stream burst correlation. Similar to [15], we represent detected bursts as a time interval of their occurrence. We provide a new burst detection scheme, which is tailored for skewed distributions, such as the financial data that we examine here. Additionally, we introduce a memory-based index structure for identification of overlapping bursts. The new index structure is based on the idea of *containment-encoded intervals* (CEI’s), which were originally used for performing stabbing queries [18]. Building on the idea of encoded time intervals, we develop a new search algorithm that can efficiently answer overlapping range queries. Moreover, we develop an approach to incrementally maintain the index as more recent data values are added. Using this new index structure we can achieve more than 3 orders of magnitude better search performance for solving the problem of burst overlap computation, compared to the B+tree solution proposed in [15]. Below we summarize the main contributions of this paper:

1. We elaborate on a flexible and robust method of burst extraction on skewed distributions.

2. We present a memory-based index structure that can store the identified burst regions of a sequence and perform very effective overlap estimation of burst regions.
3. Finally, we depict the real-time response of the proposed index and we demonstrate the intuitiveness of the matching results on financial stock data at the NYSE.

2 Problem Formulation

Let us consider a database \mathcal{D} , containing m time-series of the form $S = s_1 \dots s_n$, $s_i \in \mathbb{R}$. Fundamental is also the notion of a *burst interval* $b = [t^{start}, t^{end})$, representing a time-span of a detected burst, with an inclusive left endpoint and an exclusive right endpoint, where t^{start}, t^{end} are integers and $t^{start} < t^{end}$.

Between two burst intervals q, b one can define a time overlap operator \cap , such that:

$$q \cap b = \begin{cases} 0 & \text{if } t_q^{end} \leq t_b^{start} \\ 0 & \text{if } t_q^{start} \geq t_b^e \\ \min(t_q^{end}, t_b^{end}) - \max(t_q^{start}, t_b^{start}) & \text{otherwise} \end{cases}$$

We dissect the burst correlation problem into the following steps:

(i) Burst identification on sequences residing in a database \mathcal{D} . The burst detection process will return for each sequence S a set of burst intervals $B^S = \{b_1, \dots, b_k\}$, with a different value of k for every sequence. The set containing all burst intervals of database \mathcal{D} , is denoted as $B^{\mathcal{D}}$.

(ii) Organization of $B^{\mathcal{D}}$ in a *CEI-Overlap* index \mathcal{I} .

(iii) Discovery of overlapping bursts with a query Q given index \mathcal{I} , where Q is also a set of burst intervals: $Q = \{q_1, \dots, q_l\}$. The output of the index will be a set of intervals $V = \{v_1, \dots, v_r\}, v_j \in B^{\mathcal{D}}$ such that:

$$\sum_i \sum_j q_i \cap v_j \neq 0$$

(iv) Return of top-k matches [*optional*]. This step involves the ranking of the returned sequences based on the degree of overlap, between their respective burst intervals and the query intervals. Since this step is merely a sorting of the result set, we do not elaborate any further on this for the remaining of the paper.

3 Burst Detection

The burst detection process involves the identification of time regions in a sequence, which exhibit over-expression of a certain feature. In our setting, we consider the actual value of a sequence S as an indication of a burst. That is, if $s_i > \tau$, then time i is marked as a burst. The determination of the threshold τ depends on the distributional characteristics of the dataset. Assuming a gaussian

data distribution τ could be set as the mean value μ plus 3 times the standard deviation.

In this work we focus on financial data, therefore we first examine the distribution of their values. In Figure 2 we depict the volume distribution of traded shares for two stocks (period 2001-2004). Similar shapes were observed for the majority of stocks. We notice a highly skewed distribution that is also typically encountered in many streaming applications [1]. We capture the shape of this distribution using an exponential model, because of its simplicity and intuitiveness of the produced results. The CDF of the exponential distribution of a random variable \mathbf{X} is given by:

$$P(\mathbf{X} > x) = e^{-\lambda x}$$

where the mean value μ of \mathbf{X} is $\frac{1}{\lambda}$. Solving for x , after elementary calculations we derive at the following:

$$x = -\mu \cdot \ln(P) = -\frac{\sum_{i=1}^n s_i \cdot \ln(P)}{n}$$

In order to calculate the critical threshold above which all values are considered as bursts, we estimate the value of x by looking at the tail of the distribution, hence setting P to a very small probability, i.e. 10^{-4} . Figure 2 depicts the threshold value and the discovered bursts on two stock volume sequences.

Notice that the computed threshold is amenable to incremental computation in the case of streaming time-series (either for a sliding or aggregate window), because it only involves the maintenance of the running sum of the sequence values.

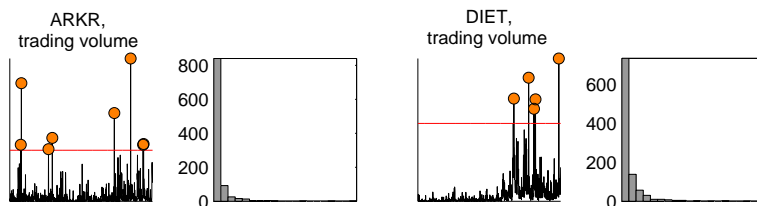


Fig. 2. Examples of the value distributions of stock trading volumes

However, setting a global threshold might introduce a bias when the range of values changes drastically within the examined window, i.e. when there is a ‘concept drift’ [8, 4]. Therefore, one can compute a variable threshold, dividing the examined data into overlapping partitions. The distribution in each partition still remains highly skewed and can be estimated by the exponential distribution, due to the self similar nature of financial data [11, 14]. An example of the modified threshold (for the second stock of Fig. 2) is shown in Figure 3, where the length of the partition is 200 and the overlap is 100. At the overlapping part, the threshold is set as the average threshold calculated by the 2 consecutive windows. We observe that in this case we can also detect the smaller burst patterns that were

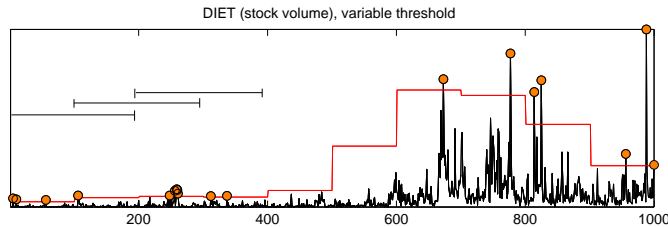


Fig. 3. Variable threshold using overlapping subwindows

overshadowed by the high threshold value of the whole window (notice that a similar algorithm can be utilized for streaming sequences).

After the bursts of a sequence are marked, each identified burst is transcribed into a burst record. Consecutive burst points are compacted into a *burst interval*, represented by its start and end position in time, such as $[m, n)$, $m < n$. Burst points at time m are therefore represented by an interval $[m, m+1)$. In what follows, we will explicate how these burst regions can be organized into an efficient index structure.

4 Index Structure

For the fast identification of overlapping burst intervals¹, we adapt the notion of containment-encoded-intervals (CEI’s), which were originally utilized for answering stabbing queries [18] (CEI-Stab). In this work we present the CEI-Overlap index, which shares a similar structure with CEI-Stab. We introduce a new efficient search technique for identifying overlapping bursts regions. Moreover, we present an effective approach for handling the nonstop progress of time.

4.1 Building a CEI-Overlap index

There are two kinds of intervals in CEI-Overlap indexing: (a) burst intervals and (b) virtual construct intervals. Burst intervals are identified as described in section 3. The notion of virtual construct intervals is also introduced for facilitating the decomposition and numbering of burst intervals, in addition to enabling an effective search operation. As noted before, burst intervals are represented by their start and end position in time and the *query search regions* are also expressed similarly.

Fig. 4 shows an example of containment-encoded intervals and their local ID labeling. Assume that the burst intervals to be indexed cover a time-span between $[0, r)^2$. First, this range is partitioned into r/L segments of length L , denoted as S_i , where $i = 0, 1, \dots, (r/L - 1)$, $L = 2^k$, and k is an integer. Note that r is assumed to be a multiple of L . In general, the longer the average length

¹ For the remainder of the paper, “burst regions” and “burst intervals” will be used interchangeably.

² Section 4.3 will describe how to handle the issue of choosing an appropriate r as time continues to advance nonstop.

of burst regions is, the larger L should be [18]. Segment S_i contains time interval $[iL, (i+1)L)$. Segment boundaries can be treated as *guiding posts*. Then, $2L - 1$ CEI's are defined for each segment as follows: (a) Define one CEI of length L , containing the entire segment; (b) Recursively define 2 CEI's by dividing a CEI into 2 halves until the length is one. For example, there are one CEI of length 8, 2 CEI's of length 4, 4 CEI's of length 2 and 8 CEI's of length one in Fig. 4.

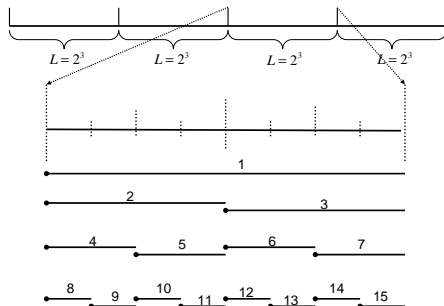


Fig. 4. Example of containment-encoded intervals and their ID labeling.

These $2L - 1$ CEI's are defined to have containment relationships among them. Every unit-length CEI is contained by a CEI of size 2, which is in turn contained by a CEI of size 4, ... and so on. The labeling of CEI's is encoded with containment relationships. The ID of a CEI has two parts: the segment ID and the local ID. The local ID assignment follows the labeling of a perfect binary tree. The global unique ID for a CEI in segment S_i , where $i = 0, 1, \dots, (r/L) - 1$, is simply computed as $l + 2iL$, where l is the local ID. The local ID of the parent of a CEI with local ID l is $\lfloor l/2 \rfloor$, and it can be efficiently computed by a logical right shift by 1 bit.

To insert a burst interval, it is first decomposed into one or more CEI's, then its ID is inserted into the ID lists associated with the decomposed CEI's. The CEI index maintains a set of burst ID lists, one for each CEI. Fig. 5 shows an example of a CEI-Overlap index. It shows the decomposition of four burst intervals: b_1, b_2, b_3 and b_4 within a specific segment containing CEI's of c_1, \dots, c_7 . b_1 completely covers the segment, and its ID is inserted into c_1 . b_2 lies within the segment and is decomposed into c_5 and c_6 , the largest CEI's that can be used for decomposition. b_3 also resides within the segment, but its right endpoint coincides with a guiding post. As a result, we can use c_3 , instead of c_6 and c_7 for decomposition. Similarly, c_2 is used to decompose b_4 . Burst IDs are inserted into the ID lists associated with the decomposed CEI's.

4.2 Identification of overlapping burst regions

To identify overlapping burst regions, we must first find the overlapping CEI's. One simple approach is to divide the input interval into multiple unit-sized CEI's

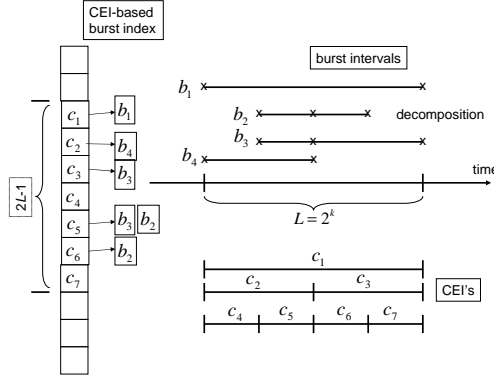


Fig. 5. Example of CEI-Overlap indexing.

and perform a point search for each of the unit-sized CEI's using the CEI-Stab search algorithm. However, replicate elimination is required to remove redundant overlapping CEI's. Fig. 6 shows an example of identifying CEI's overlapping with an input interval. There are 9 unique overlapping CEI's. Using the point search algorithm of the CEI-Stab index [18], there will be 16 overlapping CEI's, 4 from each upward-pointing dotted arrow. Seven of them are replicates. There are 4 replicates of c_1 , and two duplicates each of c_2, c_3, c_5 and c_6 , respectively, if we use the point search algorithm of CEI-Stab for searching overlap CEI's.

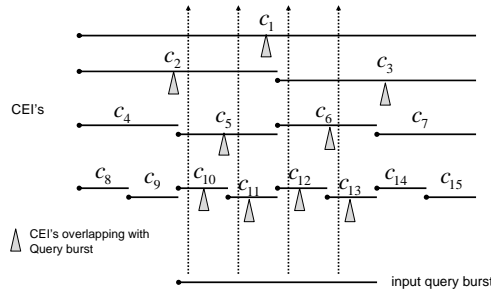


Fig. 6. Example of finding CEI's overlapping with an input interval.

Eliminating redundant CEI's slows down search time. In this paper, we develop a new search algorithm for CEI-Overlap that does not involve replicate elimination. Fig. 7 shows the pseudo code for systematically identifying all the overlapping bursts for an input region $[x, y)$, where x and y are integers, $x < y$ and $[x, y)$ resides within two consecutive guiding posts (other cases will be discussed later).

First, we compute the segment ID $i = \lfloor x/L \rfloor$. Then, the local IDs of the leftmost unit-sized CEI, $l_1 = x - iL + L$, and the rightmost unit-sized CEI, $l_2 = (y - 1) - iL + L$, that overlap with $[x, y)$ are computed. From l_1 and l_2 , we can systematically locate all the CEI's overlapping with the input interval. Any CEI's whose local ID is between l_1 and l_2 also overlaps with the input. We

```

Search ( $[x, y]$ ) { //  $[x, y]$  resides between two consecutive guiding posts
   $i = \lfloor x/L \rfloor$ ; // segment ID
   $l_1 = x - iL + L$ ; // leftmost unit-sized CEI overlapping with  $[x, y]$ 
   $l_2 = (y - 1) - iL + L$ ; // rightmost unit-sized CEI overlapping with  $[x, y]$ 
  for ( $j = 0; j \leq k; j = j + 1$ ) {
    for ( $l = l_1; l \leq l_2; l = l + 1$ ) {
       $c = 2iL + l$ ; // global ID of an overlap CEI
      if ( $IDList[c] \neq \text{NULL}$ ) { output( $IDList[c]$ ); }
       $l_1 = l_1/2$ ; // local ID of parent of  $l_1$ 
       $l_2 = l_2/2$ ; // local ID of parent of  $l_2$ 
    }
  }
}

```

Fig. 7. Pseudo code for searching overlap bursts.

then move up one level to the parents of l_1 and l_2 . This process repeats until $l_1 = l_2 = c_1$. Each overlapping CEI is examined only once. Hence, no duplicate elimination is needed. Fig. 6 shows the identification of overlapping CEI's, from which the overlapping bursts can easily be found via the CEI index.

Now we discuss the cases where the input interval does not reside within two consecutive segment boundaries. Similar to the decomposition process, the input interval can be divided along the segment boundaries. Any remnant can use the search algorithm described in Fig. 7. The full segment, if any, has all the $2L - 1$ CEI's within that segment as the overlapping CEI's.

In contrast to CEI-Stab [18], there might be duplicate burst IDs in the search results of CEI-Overlap. Note that, even though the search algorithm of CEI-Overlap has no duplicate in overlapping CEI's, it might return duplicates in overlapping burst ID's. This is because a burst can be decomposed into one or more CEI's and more than one of them can overlap with an input interval. To efficiently eliminate these duplicates, the burst ID lists are maintained so that the IDs are sorted within individual ID lists. During search, instead of reporting all the burst IDs within each overlapping CEI one CEI at a time, we first locate all the overlapping CEI's. Then, the multiple ID lists associated with these CEI's are merged to report the search result. During the merge process, duplicates can be efficiently eliminated.

4.3 Incrementally maintaining the index

Since time continues to advance nonstop, no matter what initial $[0, r)$ is chosen, current time will exceed at some point the maximal range r . Selecting a large r to cover a time-span deep in the future is not a good approach because the index storage cost will increase [18]. A better approach is to choose an r larger than the maximum window of burst regions at the moment, and to keep two indexes in memory, similar to the double-buffering concept. More specifically, we start with $[0, r)$. When time passes r , we create another index for $[r, 2r)$. When time

passes $2r$, we create an index for $[2r, 3r)$, but the index for $[0, r)$ will be likely not needed any more and can be discarded or flushed into disk. Using this approach no false dismissals are introduced, since any burst interval covering two regions can be divided along the region boundary and indexed or searched accordingly.

5 Experiments

We evaluate 3 parameters of the burst correlation scheme: (i) the quality of results (is the burst correlation useful?), (ii) the index response time (how fast can we obtain the results?), (iii) indexing scheme comparison (how much better is it than other approaches?).

5.1 Meaningfulness of Results

Our first task is to assess the quality of results obtained through the burst correlation technique. To this end, we search for burst patterns in stock trading volumes during the days before and after the 9/11 attack, with the intention of examining our hypothesis that financial and/or travel related companies might have been affected by the events. We utilize historical stock data obtained from `finance.yahoo.com` totaling 4793 stocks of length 1000, that cover the period between 2001-2004 (STOCK dataset). We use the trading *volume* of each stock as the input for the burst detection algorithm. Our burst query range is set for the dates 9/7/2001 - 9/20/2001, while we should note that the stock market did not operate for the dates between 9/11 and 9/16.

Figures 8, 9, 10 illustrate examples of several affected stocks. The graphs display the volume demand of the respective stocks, while on the top right we also enclose the stock price movement for the whole month of September (the price during the search range is depicted in thicker line style). Stocks like *'Priceline'* or *'Skywest'* which are related to traveling, experience a significant increase in selling demand, which leads to share depreciation when the stock market re-opens on Sep. 17. At the same time, the stock price of *'NICE Systems'* (a provider of air traffic control equipment) depicts a 25% increase in value. More examples of stocks with bursty trends in the stock demand within the requested time frame are presented in Table 1.

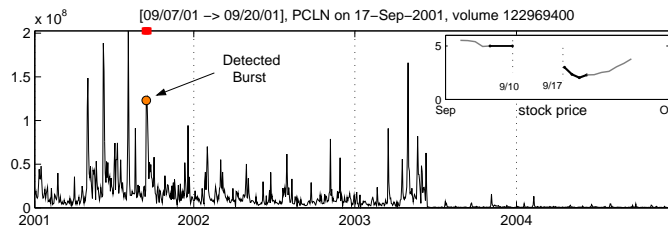


Fig. 8. Volume trading for the Priceline stock. We notice a large selling tendency, which results in a drop in the share price.

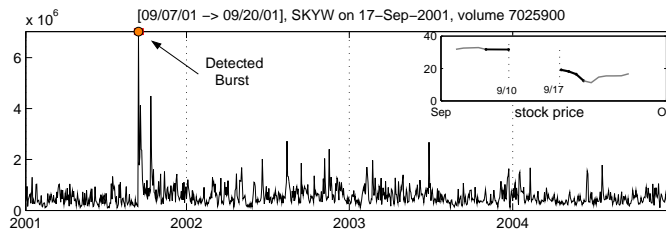


Fig. 9. Volume trading for the Skywest stock.

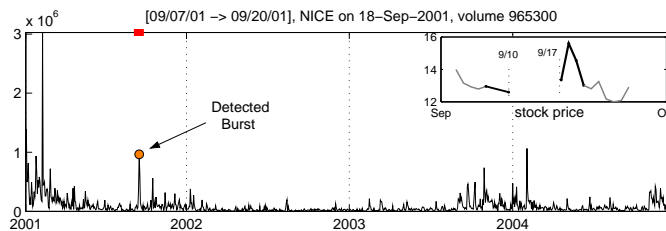


Fig. 10. Volume trading for the stock of Nice Systems (provider of air traffic control systems). In this case, the high stock demand results in an increase of the share price.

Symbol	Name (Description)	Price
LIFE	Lifeline Systems (Medical Emergency Response)	1.5% ↓
MRCY	Mercury Computer Systems	48% ↑
MAIR	Mair Holdings (Airline Subsidiary)	36% ↓
NICE	NICE Systems (Air traffic Control Systems)	25% ↑
PCLN	Priceline	60% ↓
PRCS	Praecis Pharmaceuticals	60% ↓
SKYW	Skywest Inc	61% ↓
STNR	Steiner Leisure (Spa & Fitness Services)	51% ↓

Table 1. Some of the stocks that exhibited high trading volume after the events of 9/11/2001

5.2 Index Response Time

We compare the performance of the new CEI-Overlap indexing scheme with the B+tree approach proposed in [15]. Both approaches rely on memory based indexes, so here we report the time required to identify overlapping burst regions for a number of burst query ranges. CEI-based indexing has been shown to outperform other interval indexing schemes for the stabbing case [18], such as the ‘Interval Skip Lists’ [3] and R-trees [2], therefore due to space limitations we refrain from reporting such comparisons in this version of the paper.

Because for this experiment the STOCK dataset is quite small, we generate a larger artificial dataset that simulates the burst ranges returned by a typical burst detection algorithm. The dataset contains 250,000 burst ranges, at various positions and covering different time-spans. A small sample of this dataset (cor-

responding to the bursts of 100 ‘virtual’ sequences), along with 3 query ranges, is depicted in Fig. 11. On both the CEI-Overlap and the B+tree we probed 5000 query ranges that cover different positions and ranges.

Intuitively, the cost of the search operation is proportional to the number of burst intervals that overlap with a given query. Therefore, we order the running time of each query based on the size of the answer set (more overlaps suggest longer running time). We create a histogram of the running time by dividing the range of the answer set into 20 bins and in Fig. 12 we plot the average running time of all the results that ended in the same histogram bin. The results indicate the superior performance of the CEI-based index, which is approximately 3 orders of magnitude faster than the competing B+tree approach. We should also notice that the running time is reported in $\mu secs$, which demonstrates the real-time search performance of the proposed indexing scheme.

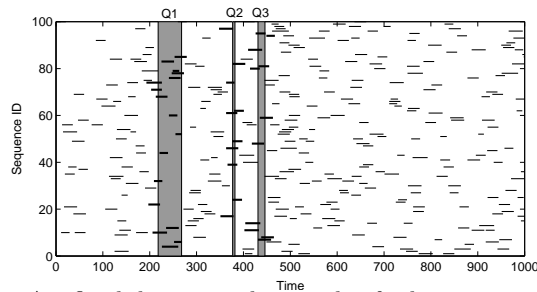


Fig. 11. Artificial dataset and example of 3 burst range queries

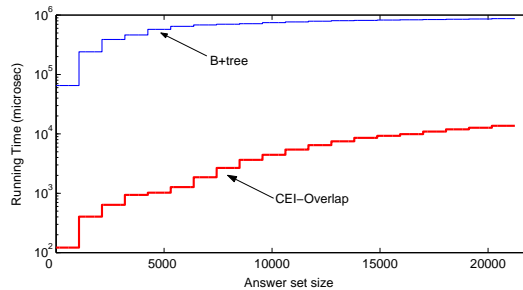


Fig. 12. B+Tree vs CEI-Overlap runtime (log plot)

6 Conclusion

We have presented a complete framework for efficient correlation of bursts. The effectiveness of our scheme is attributed not only to the effective burst detection but also to the efficient memory-based index. The index hierarchically organizes important burst features of time sequences (in the form of ‘burst segments’) and can subsequently perform very efficient overlap computation of the discovered burst regions. We have demonstrated the enhanced response time of the proposed indexing scheme, and presented interesting burst correlations that we mined from financial data. Encouraged by the excellent responsiveness and scalability of the

index, in the immediate future we plan to investigate the applicability of the indexing structure under high data rates and particularly for the online burst detection and correlation of data-streams.

References

1. G. Cormode and S. Muthukrishnan. Summarizing and Mining Skewed Data Streams. In *Proc. of SDM*, pages 44–55, 2005.
2. A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. of ACM SIGMOD*, pages 47–57, 1984.
3. E. Hanson and T. Johnson. Selection predicate indexing for active databases using interval skip lists. *Information Systems*, 21(3):269–298, 1996.
4. M. Harries and K. Horn. Detecting Concept Drift in Financial Time Series Prediction. In *8th Australian Joint Conf. on Artif. Intelligence*, pages 91–98, 1995.
5. L. J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. In *Genome Research*, 9:11, 1999.
6. H. Jiang and C. Dovrolis. Why is the Internet traffic bursty in short (sub-RTT) time scales? In *Proc. of ACM SIGMETRICS*, pages 241–252, 2005.
7. J. Kleinberg. Bursty and Hierarchical Structure in Streams. In *Proc. 8th ACM SIGKDD*, pages 91–101, 2002.
8. M. Lazarescu, S. Venkatesh, and H. H. Bui. Using Multiple Windows to Track Concept Drift. In *Intelligent Data Analysis Journal*, Vol 8(1), 2004.
9. W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic. In *Proc. of ACM SIGCOMM*, pages 183–193, 1993.
10. A. Lerner and D. Shasha. The Virtues and Challenges of Ad Hoc + Streams Querying in Finance. In *IEEE Data Engineering Bulletin*, pages 49–56, 2003.
11. T. Lux. Long-term Stochastic Dependence in Financial Prices: Evidence from the German Stock Market. In *Applied Economics Letters Vol. 3*, pages 701–706, 1996.
12. T. M. Nguyen and A. M. Tjoa. Grid-based Mobile Phone Fraud Detection System. In *Proc. of PAKM*, 2004.
13. Steven L. Scott. A Bayesian Paradigm for Designing Intrusion Detection Systems. In *Computational Statistics and Data Analysis. (special issue on Computer Security) 45*, pages 69–83, 2004.
14. A. Turiel and C. Perez-Vicente. Multifractal geometry in stock market time series. In *Physica A, vol.322*, pages 629–649, 2003.
15. M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identification of Similarities, Periodicities & Bursts for Online Search Queries. In *Proc. of SIGMOD*, 2004.
16. M.-A. Widdowson, A. Bosman, E. van Straten, M. Tinga, S. Chaves, L. van Eerden, and W. van Pelt. Automated, laboratory-based system using the Internet for disease outbreak detection, the Netherlands. In *Emerg Infect Dis 9*, 2003.
17. W.-K. Wong, A. Moore, G. Cooper, and M. Wagner. WSARE: What’s Strange About Recent Events? In *Journal of Urban Health 80*, pages 66–75, 2003.
18. K.-L. Wu, S.-K. Chen, and P. S. Yu. Interval query indexing for efficient stream processing. In *Proc. of ACM CIKM*, pages 88–97, 2004.
19. Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *Proc. of SIGKDD*, pages 336–345, 2003.