Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2014, 6(5):1169-1175

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

The research on methods of storing events detected in memory and external memory

Ya Wang and Yange Chen

Information Engineering Institute, Xuchang University, Xuchang, Henan, China

---

ABSTRACT

Technologies of event detection can help people obtain useful information from a high volume of events. Therefore people have paid more attention to them in recent years. However, with the rapid development of Data Acquisition Equipments(DAE), the number of event and event type are increasing continually, which brings a greater challenges to limited memory. By analyzing characteristics of event, we present an event model which can be applied to various fields and then two methods of storing events based on internal and external memory scheduling are proposed, which are convenient for searching events. By using of it, the events can be handled more easily due to the compression technology and more storage space can be saved. Finally, experiments on masses of events suggest that less storage space of events is used and the efficiency of event detection has been improved greatly according to the methods proposed by this paper.

Key words: Event model; tree structure; storage model in external memory

---

INTRODUCTION

The technology of event detection can discover complex events and implicit knowledge from events seeming unrelated by searching events, filtering events and relating events, and then responses can be done rapidly to hidden opportunity and thread. With the development of current RFID, sensor and other electronic data gathering equipments, masses of data have been produced and technologies of event detection have received much attention and research as a hotspot research field in recent years. So far, many products on event detection are produced, and they have been used for many fields, such as fraud detection[1], trading system[2], Business Activity Monitoring[3], medical telemonitor[4], sensor networks[5-6], multimedia[7-8], supply chain management[9-10], tracking in the library[11] etc., which brings much safety and convenience for people's life.

With the development of data gathering equipments, the type of data and the number of data are all increasing constantly. Comparing to data, the increment of memory is very limited and so far all technologies for event detection are applied totally in memory, which makes event detection for masses of data inefficient. Therefore, how to make limited memory store more events and how to improve the efficiency of event detection have become the issues most in need of solutions.

Based on situation monitoring application needs, event processing and lately data stream processing have evolved independently. Event specification languages for specifying composite events have been proposed and triggers have been successfully incorporated into relational databases. Event detection technology for event streams also has been developed, and now many detection systems are produced, such as Zstream[12], Cayuga[13], SASE[14], Sentinel[15], EVE[16], SAMOS[17] etc.. With the development of DAE, event types and event number are all increasing continually, which brings challenges to compress and store masses of data. The application of masses of data is mainly on the storage and retrieval for structured data such as amounts of log data, network message and electronic commerce and so on. Google Inc. also studies deeply on this issue and proposes the Bigtable for data management and column-oriented structure of storage, however, this method is not feasible for detecting events. The system of SASE proposes the runtime stack for recording the set of active states at a certain point and how this set leads a new set of active states as an event arrives. Each active state instance in the stack has one or two predecessor pointers specifying the active state instance that it came from, which can compress instances and improve the efficiency of detection[14]. The essay[18] proposes a kind of way to compress sensor events, however, this compression is lossy, so efficient recovery algorithms of compressed sensing is proposed to reconstruct the source signal with multiple simultaneous events. The essay[19] proposes event-based compression for data streaming which relies on the semantics of the application in driving the compression process by identifying interested events occurring in the unbounded stream. However, the above methods of compression are only feasible for events in memory and many of them are only feasible for some category of events from some field. In this article, we propose ways of compressing events both in memory and external memory and also a kind of universal event model to express events. Moreover, the two compression methods can improve the efficiency of detecting events.

STORAGE METHODS FOR EVENTS IN MEMORY AND EXTERNAL MEMORY

Event model

The current society is full of information. The manifestations for information are various, such as characters, language, graphic image, sound and so on. Among these manifestations, data is an important type and event is a kind of data. The traditional definition of event is that event is the thing which has happened or event is the condition that has changed meaningfully. In this article, event is a unit for information, which has different attributes on different occasions, and these attributes show features of the event. Attributes of event are considered as the unique identification of events, also they can be defined as the relation with other events.

Event can be used in many occasions, but for every occasion, event attributes of occurrence time and event type may be concerned most. Based on that, a kind of event model is proposed, which is shown as triple form. In this triple, there are three elements which are the object of event (ID), the type of event (ET) and the occurrence time of event (TS). The triple can be simply written as (ID, ET, TS). Generally, ET is shown by capital and event instance which means the occurrence of the event is shown by corresponding lowercase. For example, for event type A, a is the event instance, which occurs at some time. The event model by triple is universal, which can be used for different data sets.

Tree structure

The event model in this article is defined as triple of (ID, ET, TS), but in the real world, events occurring at different times can have the same attributes of ID or ET. For example, in the process of supply chain management, the mobility of the goods can be considered as an event, and the location of the goods can be considered as event type. Usually, the goods is moved to some place at some time, which means a kind of event occurs, and we can mark it as event A. The next time, the goods is moved to another place, which means a new kind of event occurs and we can mark it as event B , and after that, the goods is moved back, which means event C occurs. The objects of events A, B, C are the same, and the event types of A and C are the same. Not only for the supply chain management but that many applications are the same.

Based on the analysis above, all events that have occurred are stored in memory by triple of (ID, ET, TS), which is a waste of space. Many events are the same for the attribute of ID or ET, so we can use a kind of tree structure to compress and store events. In the tree structure, the root node can store event attribute of ID, nodes of the second layer in the tree can store event attributes of ET, and nodes of the third layer in the tree that are leaf nodes can store event attributes of TS. In one tree, the root node is only one, and nodes of the second layer and the third layer are many. One tree can store all events which have the same ID. For events that have the same ID or ET can share the same nodes for ID or ET in the root node or the second nodes, so tree structure has the function of compression.

In figure 1, horizontal axis stands for timer shaft, which shows 20 seconds as 1, 2 and so on. In this figure, we record ID for event as natural number like 1,2,3 etc., ET as capital as A, B, C etc., and the event instance as a, b, c ,etc. At every second, there has an event occurring and the attributes of ID and ET are shown as ET_{ID} above the timeline, for example, a_1 means the ID is 1 and ET is A. Occurrence times for a_1 are marked below the timeline, and they are at 1 second, 2 second, 6 second and 15 second. In 20 seconds, there are 20 events occurring with 2 kinds of ID and 3 kinds of ET, so they can be stored in 2 trees below the timer shaft.

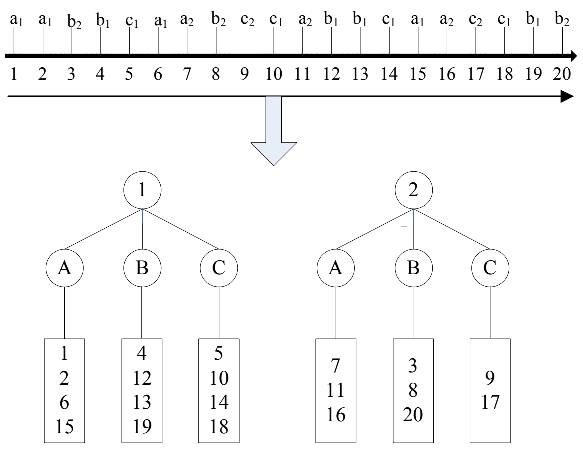

Fig.1: Tree structure is used to store events

With index, the algorithm that tree structure is used to store event instances can be described as algorithm 1.

Algorithm 1 Tree structure is used to store event instances in memory

```
Input : event instance(id, et, ts)
Output : Tree
1. search id in the root node of Tree
2. if find success
3. search et in the second layer of the Tree
4. if success
5. insert ts in the dynamic array in the leaf node of Tree
6. else
7. construct a new node in the second layer of the Tree to store et
8. construct a new leaf node of the Tree to store a dynamic array
9. insert ts in the dynamic array in the leaf node of Tree
10. else
11. construct a new Tree for id
12. store id in the root node of the Tree
13. construct a new node in the second layer of the Tree to store et
14. construct a new leaf node of the Tree to store a dynamic array
15. insert ts in the dynamic array in the leaf node of Tree
```

Event detection is often based on events with the same ID, and in the tree structure, events with the same ID are stored in the same tree, so events detected can be seek out quickly from the tree in the process of detecting, which can facilitate the detection of event and improve the efficiency of event detection to a certain degree.

THE STORAGE MODELS AS ID AND ID_ET IN EXTERNAL MEMORY

In the process of event detection, the arrival of events is streaming in, because of limited memory, many events detected can not be stored in memory momentarily, and external memory is needed. To compress events and search events effectively, events in external memory can be stored by two methods. One method is called ID, which is appropriate for event sets that have fewer event types. By ID method, events are sorted by their attributes of ID and then stored in different files which are named by ID. In ID file, only attributes of ET and TS need to be stored. For example, 20 events in figure 1 have 2 objects, and we can store them in 2 files in external memory as shown in figure 2. During the detection, if one event needs to be detected, we can only search the file named as ID of the event by index.

If ETs of events are too many, events can be stored in different files which are named as ID _ET by sorting their attributes of ID and ET. In ID_ET file, only occurrence time of event needs to be stored. During the detection, if one event needs to be detected, we can only search the file named as ID_ET of the event by index. In figure 1, 20 events which contain 2 kinds of object and 3 kinds of event type, can be stored in 6 files in external memory as shown in figure 3.

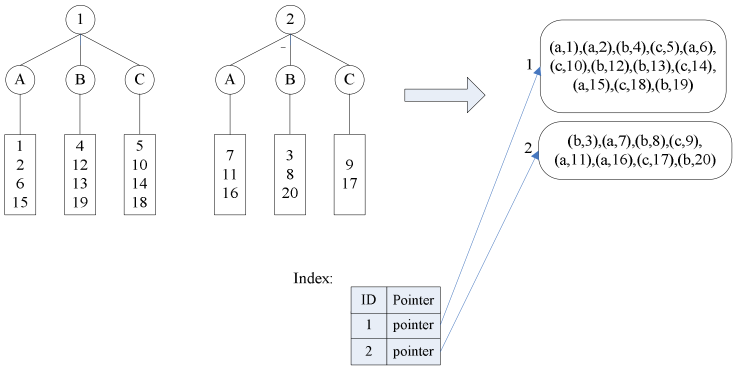
1
2
A
B
C
1 2 6 15
4 12 13 19
5 10 14 18
7 11 16
3 8 20
9 17
(a,1),(a,2),(b,4),(c,5),(a,6),(c,10),(b,12),(b,13),(c,14),(a,15),(c,18),(b,19)
(b,3),(a,7),(b,8),(c,9),(a,11),(a,16),(c,17),(b,20)
Index:
ID Pointer
1 pointer
2 pointer

Fig.2: ID files are used to store event instances

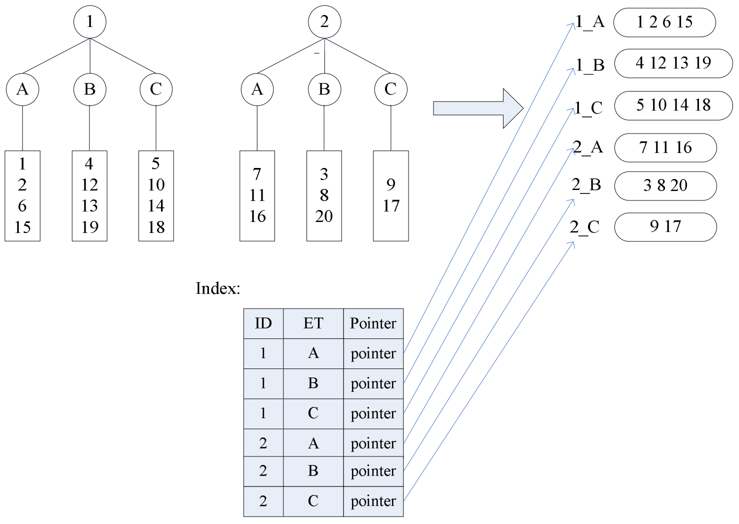

Fig.3: ID_ET files are used to store event instances

The algorithm that events are stored in external memory by sorting their ID can be described as algorithm 2.

Algorithm 2 The storage for event instance in external memory by ID file

```
Input: event instance(id, et, ts)
Output: new file or modified file in external memory
1. for(int i=1; i<=Ni; i++)
2. if Ni.id == id
3. write et, ts to file named as id
4. else
5. construct a new file named as id, write et, ts to the file
6. construct a node in index structure, write id to the node of index
```

The algorithm that events are stored in external memory by sorting their ID and ET can be described as algorithm 3.

Algorithm 3 The storage for event instance in external memory by ID_ET file

```
Input: event instance(id, et, ts)
Output: new file or modified file in external memory
  1.  for(int i=1; i<=Ni; i++)
  2.  if Ni.id == id and Ni.et=et
  3.  write ts to file named as id_et
  4.  else
  5.  construct a new file named as id_et, write ts to the file
  6.  construct a node in index structure, write id, et to the node of index
```

EXPERIMENTAL SECTION

In this article, synthetic dataset is used to prove the efficiency of methods proposed. The dataset includes 50 objects and every object includes five types of events defined as A, B, C, D, E. Events are produced by time, and 100 events occur every second. Sequence of events detected is defined as A; B; C; D; E, which means the occurrence time of the five events must satisfy sequence and all events in the sequence must occur within 1 hour. All experiments are performed on PC which has a 1.99GHz dual-core Pentium, 2.96GB of memory and 400GB of hard disk. The experimental procedures are performed with Visual C++ in windows 7 host operating system.

SPACE COST OF TREE STURCTURE

In this experiment, ways of tree structure, instance stack of SASE system and without any compression are used to store event instances produced in different times. Space costs for storing the same instances are compared to distinguish advantages and disadvantages of the three storage methods. The experimental result is shown in figure 4.

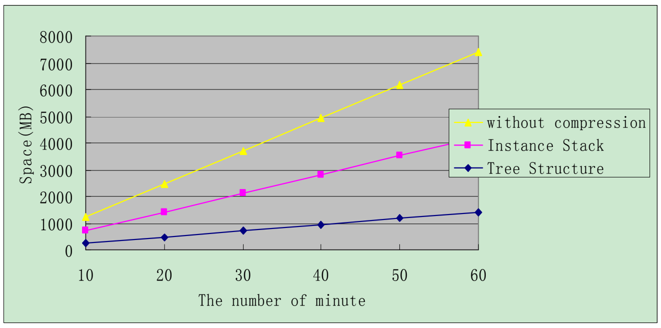

Fig.4: Comparison on space costs

In figure 4, horizontal axis shows time, and vertical axis shows space costs storing events produced in the same period of time by three storage ways. With time increasing, event instances are produced more and more, and it can be seen that space costs for three methods are increasing continually. For the method of storing instances without any compression, space cost relies on linear growth with time increasing, because the number of event instances is increasing linearly by time. For tree structure, only occurrence time of every event must need to be stored, and events with the same ID or the same ET can share the nodes of ID or ET in the tree, so tree structure saves much space. For instance stack for SASE system, many pointers are used to link related events for detecting expediently, so with the number of events increasing, the number of pointer is increasing, and the space cost is also increasing. From the figure, we can see that tree structure storing the same number of events costs least space, which means tree structure has the function of compression and it can make limited memory store more events.

DECECTION TIME COST OF TREE STURCTURE

In this experiment, we suppose that all events produced in 1 hour can be stored in memory. Event detection is used by tree matching model. The model based on tree matching relies on full pattern matching when it is used to detect complex events, which means that corresponding tree should be constructed for every event in the event pattern. When some basic event is coming, relevant leaf node will be informed, and then the leaf node will send message to its father node. The father node will use the records of its new child node and the records of other child nodes to produce possible new event records according to semantic meaning, and then the father node will inform its father node. This process will continue until complex events are detected or new records will not be produced[15]. Figure 5 shows time costs for detecting events stored by tree structure and without any compression structure. From the figure, we can see that the time cost for detecting the same events stored by tree structure is less than time cost of events stored without compression structure, because events detected often have the same object. For tree structure, events with the same object can be stored in one tree which can facilitate to search events detected.

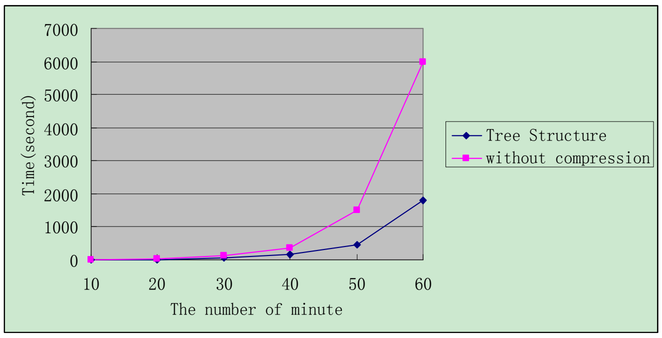

Fig.5: Comparison on time costs

SPACE COSTS OF ID FILE AND ID_ET FILE

In figure 6, horizontal axis shows time, and vertical axis shows space costs to store events produced in the same period of time by ID file, ID_ET file and without any structure. From the figure, we can see that the way of ID_ET storing the same events costs least space and the way of without any structure storing the same events costs maximum space. It is because that for the way of ID, the attributes of ET and TS must be stored in the file. For the way of ID_ET, events that have the same ID and ET can simply store occurrence time of the event, and for the way of without any structure, three attributes of every event must be stored. Comparing with the other two ways for storing events, storage without any structure costs space maximum, which means that the two ways of storing events proposed by this article have the function of compression, and they can save space in external memory.

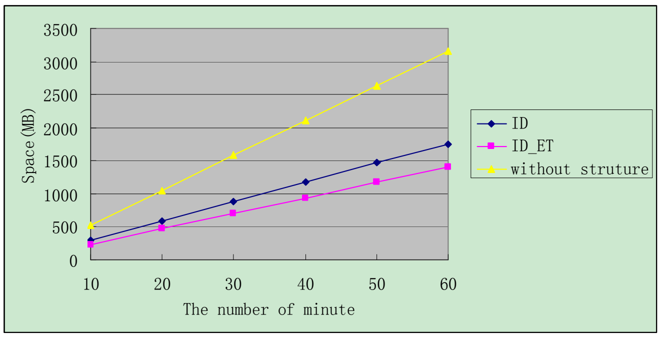

Fig.6: Comparison on space costs

DETECTION TIME COSTS OF ID FILE AND ID_ET FILE

In this experiment, we suppose that events detected cannot be stored in memory totally and the memory can only store events produced in 10 minutes. Event detection is used by tree matching model. In the process of detection, when the memory is full, the coming events will be stored in external memory and wait for being detected. In figure 7, horizontal axis shows events produced by time, and vertical axis shows time costs for detecting events stored by three ways in external memory. From the figure, we can see that detection time is increasing with the time increasing, because the number of events detected is increasing with time. For detecting the same events produced in the same period of time, the storage method for events stored by ID_ET costs least detection time and the method of events stored without any compression structure costs maximum time, which means the storage methods of ID and ID_ET can improve the detection effectively. That's because events detected often have the same object, events of the same object can be quickly found in the file ID_ET or ID, and events without being stored by classify will be hard to find them quickly. Comparison with events stored by ID, events stored by ID_ET provides more accurate location to search events detected, so the detection time is less.

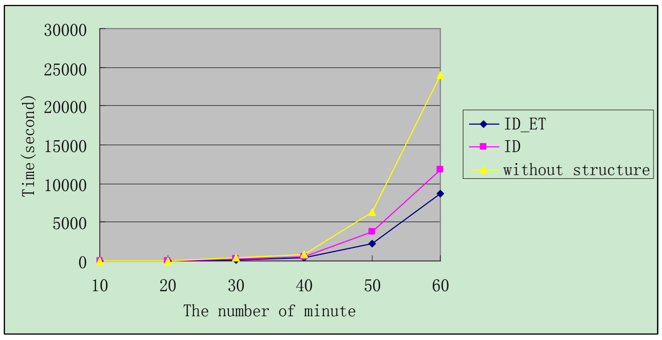

Fig.7. Comparison on time costs

CONCLUSION

With the development of data gathering equipments, the number and the type of events are increasing all the time, which brings a great challenge for limited memory in the process of event detection. This article firstly proposes a universal event model which can be applied to kinds of fields, and then proposes tree structure and the storage ways by ID and ID_ET to compress and store events in memory and external memory. Finally, experiments prove that tree structure and storage ways by ID and ID_ET can compress events and make event detection more efficient.

Acknowledgments
The authors wish to thank the Science and Technology Planned Project of Xuchang (Grant No:5017) and key Project of Natural Science of Henan Education Department(Grant No:14A510018), under which the present work was possible.

REFERENCES

[1] Angell I., Kietmann J.. *Communications of the ACM (CACM),* **2006**,49(12),91-96.
[2] Al M., Simson G., Beth R.. *RFID: applications, security, and privacy*. America:Addison-Wesley, **2006**,33-47.
[3] Vu C, Beyah R, Li Y S. *Proc.of IPCCC,* New Orleans,LA, **2007**, 264-271.
[4]Phuong Nguyen Cong, Dat Tran Do. *Computing, Management and Telecommunications (ComManTel),* Ho Chi Minh City,Vietnam,**2013**, 330-333.
[5]Keally M, Gang Zhou, Guoliang Xing. *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Stockholm, Sweden, **2010**, 279-288.
[6]Lian Hao,Chen Xinrong. *Electric Power Automation Equipment*,**2010**,9(9),125-128.
[7]Zhigang Ma, Yi Yang, Nicu Sebe, Kai Zheng. *IEEE Transactions on Multimedia*, **2013,** 15(7):1628 – 1637.
[8]Wang Hongguang,Zhou Yi,Tong Mingqiang, el al.*Journal of Tianjin University of Technology*,**2006**,22(5):21-24.
[9]Carzaniga A. , Wolf A.L. *SIGCOMM*, New York, **2003**, 163-174.
[10]Li Han,Wang Ku,Liu Shaojun. *Power System Protection and Control*,**2011**,39(11):111-115
[11]Rizvi S., Jessery S.R., Krishnamurthy, S., Franklin, M.J., Burkhart, N. et al. *SIGMOD*, New York, June **2005**,885-887.
[12] Mei Y., Madden S.. *ACM SIGMOD Conference on Management of Data,* New York, June **2009**, 1-14.
[13] Demers A., Gehrke J., Panda B., Riedewald M., Sharma V., White W.. *3rd Biennial Conference on Innovative Data Systems Research (CIDR'07),* Asilomar, CA, USA, January **2007**, 412-422.
[14] Wu E., Diao Y., Rizvi S.. *ACM SIGMOD Conference on Management of Data*, New York, **2006**, 407-418.
[15] Chakravarthy S., Mishra D.. *Data Knowledge Engineering*,March **1994**,14(1), 1-26.
[16] Geppert A., Tombros D.. *IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*， New York, **1998**, 427-442.
[17] Gatziu S S., Dittrich K. R.. *IEEE Bulletin of the TC on Data Engineering,* December **1992**, 15(1-4),23-26.
[18] Xuqi Zhu ,Cong Ma,Lin Zhang. *18th International Conference on Telecommunications (ICT),* Ayia Napa,Cyprus, May **2011**, 27-32.
[19]Alfredo Cuzzocrea, Sharma Chakravarthy. *12th International Conference KES, Zagreb, Croatia*, September **2008**, 51(78),670-681.