Concurrent Automata, Database Computers, and Security: A "New" Security Paradigm for Secure Parallel Processing

T. Y. Lin
Department of Mathematics and Computer Science
San Jose State University
San Jose, California 95192

Abstract

Declustering has been proposed to speed up parallel database machine. However, the security requires clustering. In this paper, we use temporal clustering to reconcile the apparent conflict. Automata theory is applied to high level architecture design. Based on Petri net theory a database machine is proposed. The classical notion of clustering is extend to temporal dimension and is imported to parallel database systems. The proposed database machine not only has the linear speedup, its capability (modeling power) also is increased in the order of magnitude. The computational model (in terms of automata) of the total system is strictly higher than the union of that of individual machine. It also efficiently support the security.

1 Introduction

In this paper we continue our efforts on applying Petri nets and automata theory to issues in parallel architecture [Lin90a, 91a]. The main focus here is in database computers and their security.

Security has some architectural implications. We have shown that a 'multilevel data model' without additional structure can not be secure in the sense of Bell and LaPadula Model (BLM) up to the letter; in a very strict sense, there are 'flaws' in current interpretations of BLM. We have proposed a new secure model in which the new architectural requirements is integrated into the multilevel data model [Lin92g]. Here we are applying this new model to parallel database systems.

In classical databases, the notion of clustering is employed to speed up retrieval. In secure databases, clustering is employed to protect our data. So **clustering is an essential notion in both ordinary and secure databases**. Clustering in its primitive form seems in conflict with parallel processing. **Declustering has been proposed as a means of speeding up parallel processing** [DeWitt91]. In this paper, we propose a "new" security paradigm for parallel database systems. The essence is to reconcile this apparent contradictory requirements between security and parallel processing.

Traditionally, clustering means that related data are stored together physically. The advantage of it is that in one access, one can retrieve related data into the buffer so that there is no need to access secondary storage again for related data. This strategy, in parallel systems, in fact has negative effects. So the notion of declustering surfaced. We propose, instead of declustering, but to **redefine the clustering in temporal dimension**. The notion of clustering though originated from temporal, not physical, requirements. However, in the past the notion of clustering has been taken literally, because the two requirements agree. In this paper, we propose to use temporal proximity as our 'new' paradigm for clustering which will bring the notion of security and parallel processing harmonily together. The 'new' clustering makes most of classical concepts portable to parallel processing world.

Based on previous studies of concurrent automata, we are developing a new type of parallel database machine, called EPN-DB (Extended Petri Net Database machine). A reformulated ABDM is then mapped onto the EPN-DB Machine. Based on the new notion of clustering, we found that we can simply port the classical requirements into parallel database machines without introducing declustering. At the same time security's requirements on the storage structure can also be easily ported into our EPN-DB. We will show that our new EPN-DB is not only an efficient parallel processor also a very secure one.

*This work is partially supported by the grant MDA904-91-C-7048.

2 High Level Computer Architecture

It may worthwhile to recall some of our fundamental views here. Computer architecture normally is defined as assembly language programmers' perception of hardwares [Brink87, p. 6]. We are exploring the perception of the algorithm designers or software engineers [Lin91a]. In the extreme, Church thesis is one form of computer architecture, because it represents computer scientists' perception of hardwares. To us, Church thesis captures the maximal capability of hardwares. While our approach is to capture the practical, normal, or natural capability of computers. This approach is inherently controversial. Words such as "normal," or "natural" means different things for different people. However, we believe that this approach has some practical usages. Our motivations come from two sources: our construction of concurrent automata as a solution to Peterson conjecture [Lin90a] and Tanenbaum's comments on the organization of the stack of IBM PC, see [Tann87, pp.226] and 3.1 below.

We view automata as abstractions of hardwares, as well as abstractions of algorithm types or procedure types. Our scenario is as follows: We analyze the desirable systems and identify the algorithms types (in terms of automata) to be supported. Then we examine the hardwares of the automata. For example, if the algorithm is simulatable by a pushdown automaton then we need a hardware, called pushdown automaton machine (PDA-machine). We have identified earlier that microprocessors with stack overflow exception are PDA-machines [Lin91a]. The Intel 80i86 (i = 2, 3, ...) are such machines (see the previous Proposition). Applying the scenario, we build a new type of database computers, EPN-DB, which is not only efficient and can be organized, via new notion of clustering, to support the security requirements in parallel environment.

3 Extended Petri Net (EPN) Parallel Architecture

In this section, we review, motivate, and generalize our approaches to Petri nets and EPN parallel architecture [Lin90a, 91a].

3.1 Peterson Conjecture

Petri nets have been proved to be one of the powerful models for parallel and concurrent systems. Many applications and generalization have been developed in past decades [Mura89]. Interestingly, many of these generalizations turned out to be either equivalent to Petri nets (not real extension) or equivalent to Turing machines (too much extension). For examples, color Petri nets are generalizations, yet equivalent to ordinary Petri nets [Jens81], [Vos80]. On the other hand, Petri nets with inhibitor arcs are real, but too much extensions; they are Turing machines [Agew74]. These phenomena led James Peterson and many others to believe that **"Any significant extension of the Petri net model tends to be equivalent to a Turing machine"** (Computing Surveys [Pet77, pp. 249] and his book [Pet81, pp. 203]).

A class of Extended Petri nets (EPN), called Concurrent Pushdown Automata (CPDA), was constructed in [Lin90a]. In terms of modeling power, CPDA are strictly below Turing machines and strictly above (ordinary) Petri nets. The existence of CPDA settled Peterson's conjecture negatively. The concurrent automata are the backbone of our studies. CPDA is an ordinary Petri net augmented with pushdown automata. The augmentation proceeds as follows:

1. At each transition, we place a PushDown Automaton (PDA) to process the color of a token that passes through the transition (color is a string of input alphabet).
2. The PDA scans and accepts a (nonempty) prefix of the color and sends the token with new color (the remaining string) to output places.
3. The resulting automaton is called **extended Petri net or concurrent pushdown automaton**. Instead of PDA, we can augment any automaton, such as finite automaton, linear bound automaton, and Turing machine at each transition of a Perti net; we shall call them concurrent automata. These concurrent automata are extensions and concurrent version of CLASSICAL automata and Petri nets.

We believe these concurrent automata will be useful in parallel processing. Intuitively, each individual automaton represents an individual computing machine and the Petri net represents a concurrent control or a network system. So concurrent automata can be viewed as computation models of parallel systems or networks. The concurrent pushdown automata that solve the Peterson's conjecture suggest that the computational power of the total system can be **strictly greater than** the union ("linear sum") of each individual system in the order of **magnitude**. So the par-

allel systems which are designed under these concurrent automata should exhibit a major jump on computing capability; it is better than liner scaleup.

3.2 Church Thesis and Natural Computation Model (NCM)

Church thesis asserts that computers can execute any algorithm, or equivalently, computers are hardware realization of Turing machines. However, a particular computing hardware may be able to execute any algorithm, it may not be very **natural, practical**, or **efficient** to execute some type of algorithms. So we propose to capture the naturality, practicality, or efficiency of hardware usage in the **Natural Computation Model** (NCM). NCM is a model for practical or natural capability of a given computer, while Church thesis is a model of the theoretical maximum capability.

Tanenbaum stated that "Intel 8088's memory management architecture is very primitive ...does not even detect stack overflow, a defect that has major implications ..." [Tann87, pp.226]. If machines have no stack overflow detection, designers are forced to design their software based on the assumption that the stack is of finite length. This remark leads us to conclude that IBM XT model **cannot be used naturally or practically** as Turing machines.

Proposition 1 *The NCM of IBM PC (XT model) or Intel 8088 microprocessor is a finite automata (FA), and the NCM of IBM AT (286, 386 and above models) is a PushDown Automata (PDA).*

Remark: Without explicitly abstraction of hardwares, this is not a provable proposition; it is a hypothesis. However, one can easily give an IBM PC version of RAM machine [Aho74].

For each hardwares there are NCMs associated with it. To emphasize its functionality, we call these hardwares NCM-machines, e.g., Finite Automata machines (FA-machines) PushDown Automata machines (PDA-machines), Petri Net machines (PN-machines), or Extended Petri Net Machines (EPN-machines) [Lin90a].

3.3 EPN Parallel Architecture

In parallel processing, the common concern is computing speed. Here our main focus is on the computing capability: Is there a superlinear increasing in computing (modeling) power? The answer is 'yes' for EPN parallel processing. Intuitively, modeling is the capability of solving problems. For example, finite automata can solve problems which are equivalent to some regular languages; no finite automata can parse a genuine context free languages. So the question, "Is the NCM of EPN-machine has superlinear increased in modeling power?" can be rephrased in terms of automata theory, "Is the modeling power of NCM of total system is strictly greater than the 'union' of the NCMs of individual components?" The answer is essentially given in my solution of Peterson's conjecture [Lin90a], [Lin91a].

The EPN-machines are somewhat similar to connection machines. The Petri net control can be performed by a conventional machine (host). In connection machine, the PE (processing element) is a small FA-machine. EPN-machine requires each PE to be a genuine PDA that is a microprocessor with stack overflow exception. The computing power of an EPN-machine is strictly greater than that of PDA-machine and PN-machine. In other words, the EPN-machine can solve problems which are beyond the problems solvable by PDA-machines or PN-machines. Since we are using conventional machine to support the Petri net control, we could use more powerful automaton than Petri net as its control system. For example, we could choose Petri nets with priority (equivalent to Turing machine) as its control system. For convenience the host machine is called **control machine**. Such generalization still will be called EPN architecture.

Circles and squares represent places and transitions.

4 Parallel Processing and Clustering

Common secondary storage structures for parallel database systems are declustering [Livn87], [DeWit91]. One of the key reasoning for using declustering in a parallel database is to enable the system to reading or writing multiple data stores in parallel. There are three approaches in declustering; range partitioning, round robin, and hashing. **Traditionally, clustering means that related data are stored together physically**. The advantage of it is that in one access, one can retrieve related data into the buffer so that there is no need to access secondary storage again for related data. This strategy, in parallel systems, in fact slow down the process. We propose, instead of declustering, to **redefine the clustering in temporal dimension**, and proceed as conventional databases.

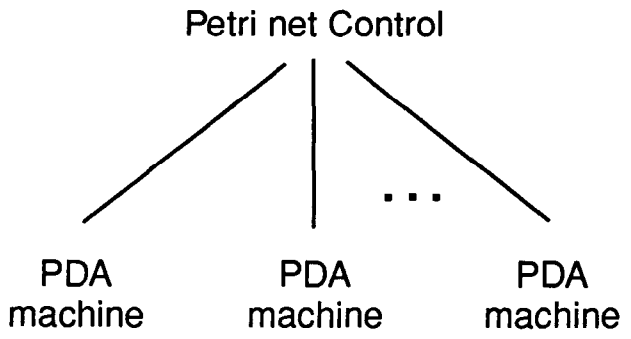

Figure 1: Petri net control of PDA machines

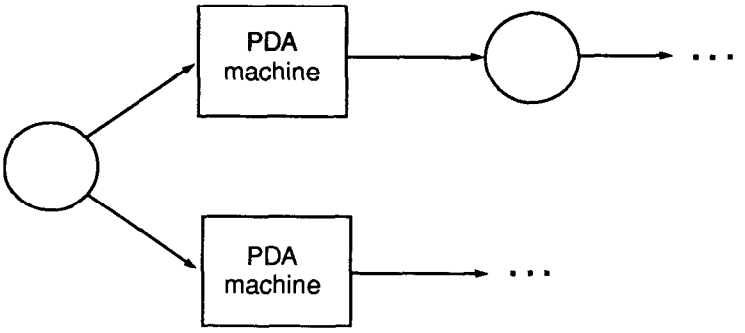

Figure 2: An EPN machine

Temporal Clustering

Let us use 'block' as the unit of data that can be retrieved by one secondary storage access. In uniprocessor world, related data are stored block by block consecutively (physically), in hoping that related data would be retrieved by one or very few accesses. To achieve this in parallel processing world, the storage organization needs to be restructured. We mimic the classical clustering almost literally, but on the time dimension. Namely, block by block, the data are evenly distributed through different data stores so that they can be retrieved in one parallel access. We illustrate our idea in PDA stores. We will call the total block from **Block #1** to **Block #n** the temporal block #1, the total block from **Block #(n+1)** to **Block #2n** the temporal block #2, and so on. The whole temporal block can be retrieved to the buffers in one parallel access. So a temporal block is something equivalent to a classical block. The net effect of this approach is that each classical database action is parallelized. The data of a temporal block is n times larger than a classical block, but it can be retrieved in one classical block time. So our organization is **n time linear speedup**. For example, NCUBE, can connect to 1024 diskdrives, so its temporal block is 1K bigger than classical block, yet it can process the data in one classical block time. On the surface, a temporal block is similar to range partitioning or round-robin declustering. Intrinsically, temporal blocks are distributed by the semantic of data, so the distribution is more nature and effective.

5 EPN Database Architecture

In this section, EPN-architecture is applied to databases. In order for EPN-architecture to be useful in databases, some data stores are attached;they are called EPN-DB.

5.1 ABDM and EPN-DataBase Machines

An EPN-DataBase machine (EPN-DB machine) is an EPN-machine attached with secondary storage structures to support data models. To each PDA-machine, we attach a secondary data storage, called PDA-store. To be specific PDA-machines are called PDA-processors. The pair is called PDA-database machine (PDA-DB). Optionally, we may attach a sec-

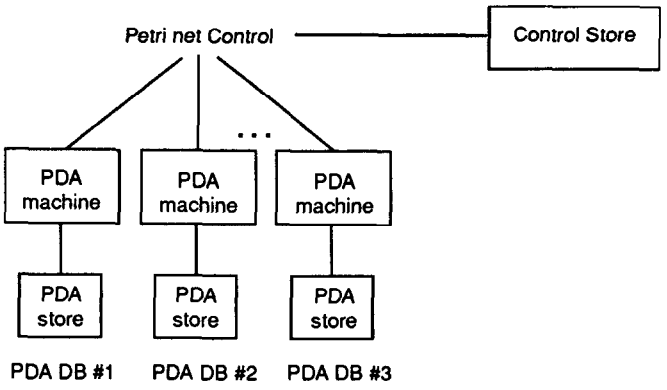

Figure 4: Petri net control of PDA machines, stores, and databases

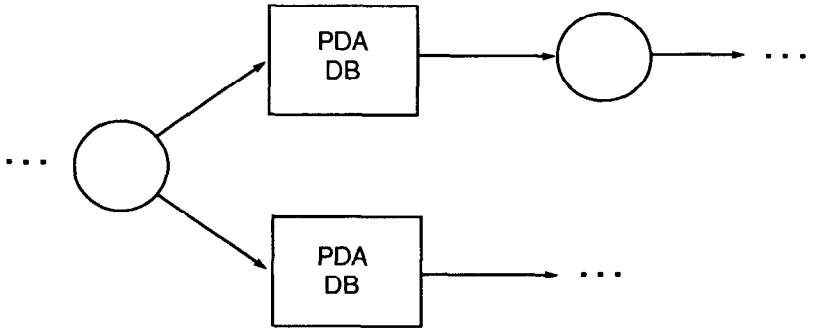

Figure 5: EPN database machine

Example 5.2.

At the user's level, we can represent ABDM in relational form (There are intrinsic difference, see [Lin92h]). So we will borrow the SQL for illustration. Suppose a SELECT is issued. The command is sent to each PDA-DB, and evaluated on the MetaData. Based on the information on MetaData, the desirable data is retrieved from BaseData.

Suppose the given SELECT is as follows:

```
select TARGET1, .....,TARGETING
from TABLE1, ....TABLEAUX
where TABLE1.ATTRIBUTE1= VALUE1
and TABLE1.ATTRIBUTE2=TABLE2.ATTRIBUTE2 (1)
and ...(2)
```

The induced SELECT, denoted by [SELECT], is as follows: the MetaData and their data are represented by []:

```
select [TARGET1],.......,[TARGETING],
from [TABLE1],.....[TABLEAUX]
where [TABLE1].[ATTRIBUTE1]=[VALUE1] [1]
and [TABLE1].[ATTRIBUTE2]=[TABLE2].[ATTRIBUTE2] [2]
and ...
```

MetaData's are small, each PDA-DB can evaluate the derived query [SELECT] on its own. Each then locates the clusters that contain [TARGET1], ...[TARGETING] and the variable /data which are in the predicates [1], [2], ...Since the number of clusters are then very small, so the original SELECT can be evaluated by all PDA-DB's coordinated by Petri Net control. Thus, the TARGET1, ...TARGETING can be retrieved most efficiently and quickly.

5.2 Realization of EPN-DB

We have not imposed any constraints on the Petri net yet, so the EPN-DB represent a very wide ranges of computer systems. Our illustrations have been concentrated on share nothing architecture, because it is most useful and popular. The EPN-DB can easily represent other architectures, such as the share main memory architecture or share disk architecture. Each PDA-DB can be a microprocessor based computers, such as IBM PC (386 and higher models), work stations. The control machine can be a mainframe or another microprocessor based computer. The machines are linked by LAN. Many multimachine are expressible in EPN-DB. We believe EPN's are good theoretical "shell" for high level architecture design.

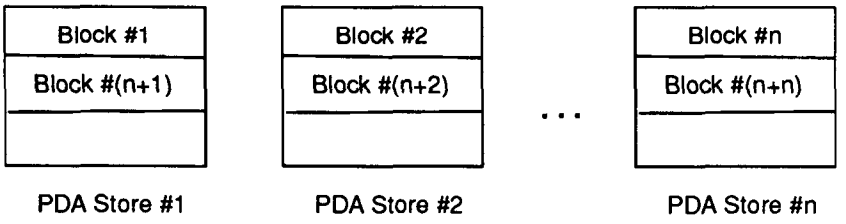

Figure 3: Temporal clustering

ondary storage to the control-machine (Petri Net-machine). We will call them control-processor and control-store. Such architecture is called EPN-DB architecture. To claim that EPN-DB are database machines, we have to demonstrate that they do support data models. We will map ABDM to EPN-DB.

The Storage Structure of BaseData

The Basedata should be stored in data stores. ABDM requires that all related tuples are stored in a cluster. We interpret clusters in ABDM as temporal clusters. Therefore an ABDM cluster in parallel processing is a temporal block, so its data are distributed evenly into each PDA-store, as we have illustrated in the temporal clustering. Each PDA-DB handles ideally only one block for each retrieval. That is, EPN-DB handles one temporal block for each parallel retrieval, at worst very few temporal blocks. Then conventional strategies are portable to temporal blocks, and we do not need the insecure declustering.

The Storage Structure of MetaData

MetaData is much more smaller than the size of BaseData, so we have following choices:

1. It is resided in the control machine if the control machine is a mainframe computer.

2. If the control machine is smaller, we replicate the MetaData into each PDA-DB. In this way, each PDA-processor can access their respective MetaData store parallelly.

EPN-DB includes the so called share nothing architecture [Ston88]. PDA-stores do not share anything. PDA-processors communicate among themselves via Petri net control. When a user issues a request, the Petri net control puts tokens into proper places(to initiate PDA-processors), and then run EPN.

Circles and squares represent places and transitions.

Example 5.1.

Let us illustrate our scenario on a very specific example, namely, a search on a clustered index (the index order is the physical storage order)[Date90]. Here the clustered index is interpreted as the index clustered in temporal dimension. Namely, the index should be line up from block #1, block #2, ...until the whole temporal block is filled. Then it goes to second temporal block and etc. EPN-DB is a Petri net. Each PDA-DB is a subnet in which PDA-processor represents a **transition** and PDA-Store a **place**. When a user requests for searching an index. A token with the desirable color (index) is placed in each of the "little" place displayed in the figure. If the color(source key) matches the color (the index) in PDA-Store, the particular PDA fires. The clustered index is searched in every PDA-store, if "match" the PDA "fires" the related data identified by the index to Control machine. The token is then placed in the oval shaped output place. This token is then available to the user. In general all the classical strategies should work as long as we interpret the classical cluster as a temporal cluster.

Before we give a more detail example, let us examine our database language. Assume, we have a predicate P(x1, x2, ...) define on BaseData. Let X be a sub-BaseData in which P(x1, x2, ...) is valid. Let [X] represents the image of X in MetaData, which is the quotient set of the BaseData under the equivalence relation. Let [P] ([x1], [x2], ...) be the "induced" predicate, namely, [P] ([x1], [x2], ...) is true iff there exists xi in [xi] such that p(x1, x2, ...) is true.

Note that we are not saying that P(y1, y2, ...) is valid for any choices of yi in [xi]. We are saying that [P] is true iff there is one set of choices so that P is true. We will vaguely refer to this induced language as "quotient language" on MetaData. See [Lin92a] for the formal theory of quotient language.

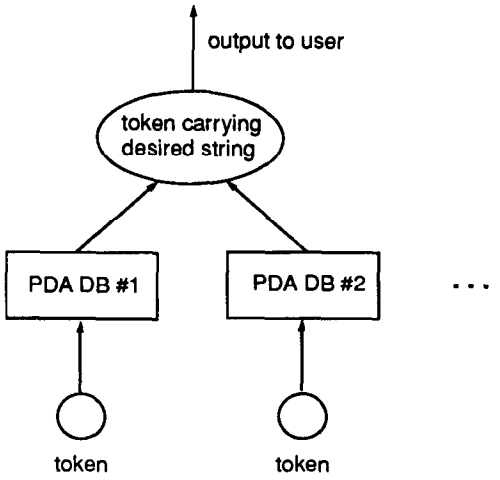

Figure 6: Search on a clustered index

6 Secure EPN-DB Machine

[DeWitt91,92] suggested that declustering is a good strategy to speed up parallel processing. On the other hand, security requires clustering. The purpose of this section is to reconcile these two apparently contradictory requirements in parallel processing.

6.1 Secure Clustering

In [Lin91a], [Lin92f], we have shown that the security level of an input to any relational operator always dominates that of the ouptput. In other words, relational operators send high level information downward to equal or lower level information. Without proper additional structure, all relational operators are trusted subjects; this is unacceptable. So in [Lin92g] we added storage requirement to the axiom of BLDM. With such requirement, one can avoid trusted subjects completely. Such storage technique, called clustering, was developed , not for security, by Hsiao in the early 70's for his model [Hsiao70], Wong71].

The notion of clustering is based on partitioning of attribute domain. Consider the following partition:

Q11 = {SALARY: 0 ≤ SALARY ≤ 40,000}
Q12 = {SALARY: 40,000 < SALARY ≤ 100,000}
Q13 = {SALARY: 100,000 < SALARY ≤ 250,000}
Q14 = {SALARY: 250,000 < SALARY ≤ 1,000,000}
⋮

It is clear that the partition defines an equivalence relation on the SALARY figures. Using ABDM, we will define an equivalence relation, called secure clustering equivalence, as follows: Two data are secure clustering equivalence iff they are in (1) the same partition, and (2) the same security class. The secure clustering equivalence relation partitions the attribute domain into clustering equivalence classes, such class is called a secure cluster. The storage requirement is:

Axiom 1 *Data Clustering Axiom: Each primitive data belongs to one and only one cluster, and each cluster has to be stored physically together. Different classes of data are stored in different volumes.*

Definition 1 *An ABDM with secure clustering equivalence satisfying data clustering axiom is call SABDM(Secure Attribute Based Data Model).*

As an illustration, suppose a user is viewing a relation, named HighView, at his terminal. We assume HighView is classified SECRET. Now if the user issues a query to retrieve a sub-relation, named LowView FROM HighView. The subrelation LowView is classified CONFIDENTIAL. Effectively, he is defining another relation LowView. Although it appears as if the data in HighView is flowed into LowView, because of the data clustering, there are no actual data flow from HighView to LowView. The LowView get its data from proper clusters, not from HighView. So such a clustering technique allows us to execute all relational operators without using any trusted subjects.

$$HighView \longrightarrow Clusters containing data in HighView \quad (1)$$

$$LowView \longleftarrow Clusters containing data in LowView \quad (2)$$

In general, we will call such data organization secure clustering. Secure clustering is secure and efficient [Hsiao88]. Since clustering is so intrinsic to security, declustering is not a secure strategy. However, in the next section, we will show the solution to this apparent contradiction.

6.2 Secure Temporal Clustering

A temporal block should have same security classification. Earlier we have pointed out that in parallel computation, the clustering should be viewed from temporal dimension. Each classical cluster is assigned one security level and receives only the data of that level, in parallel environment, temporal clusters should play the role of clusters.

6.3 Secure Parallel Data Organization

Each PDA-DB has one PDA-processor and a PDA-store. Each PDA-store consists of a set of diskpacks; in PC's, they are hardisks. A block is the collection of data which can be retrieved in one access. In 360K double density double side diskette, a block is a cluster (two physical sectors, 1024 bytes); in 1.2MB high density diskette, it is a sector(512 bytes). To be specific, let us assume that a **block is a track**. A collection of tracks with the "same address" in each different PDA-store is a temporal block. Each PDA-processor accesses only one block in one parallel access; however, in total the EPN-DB accesses the whole temporal block. The whole temporal block is assigned one security label.

Let us illustrate our idea in the organization of clustered index (the index order is the physical order in storage)[Date90]. The data will be arranged in the index order throughout the whole track. If the first track of the first PDA-DB is full, then go on to the second PDA-DB and store the data in the track which has "same address" in the PDA-store of second PDA-DB. We continue in this fashion until the whole temporal block is full. Then go on to the second temporal block, and etc. The whole temporal block is full with the data of the same security, even though, many of them are resided in different physical volumes—we, however, could say that the data are in the same "parallel" or "temporal" volume. Similarly, we can define the "parallel" or "temporal" cylinders (= the "same" tracks in difference surfaces[ElNa89,p67]). With such "parallel" or "temporal" tracks, cylinders, or volumes, the classical database can be ported into parallel processing world. So an EPN-DB machines is a secure parallel database system if one uses the "parallel" or "temporal" secondary storage structures as organized above.

6.4 Secure Relational Operations

The secure data organization give us the security, we illustrate the operations by examples.

Example 6.2.

Let us reinterpret Example 5.2. It will be seen that security requirements have no essential impact on the query processing. The security requirement is reflected in the clustering equivalence and that's all.

Suppose the following query is issued:

```
Q: select TARGET1, ...,TARGETING
from TABLE1, ...TABLEAUX
where TABLE1.ATTRIBUTE1 = VALUE1 (1)
and TABLE1.ATTRIBUTE2 = TABLE2.ATTRIBUTE2 (2)
and ...
```

Then, Q is modified into [Q] by control machine or each individual PDA-DB depending on the organization (see Section 5.1). Each PDA-DB has a copy of Q and [Q] : The MetaData and their data are represented by []

```
[Q]: select [TARGET1], ...,[TARGETING],
from [TABLE1], ...[TABLEAUX]
where [TABLE1].[ATTRIBUTE1] = [VALUE1] [1a]
[TABLE1].[CLASS] ≤ Low [1c]
and [TABLE1].[ATTRIBUTE2] = [TABLE2].[ATTRIBUTE2] [2a]
[TABLE1].[CLASS2] = [TABLE2].[CLASS2] [2c]
and ...
```

MetaData's are small, each PDA-DB can evaluate the derived query [Q] on its own, and locates the clusters that contain [TARGET1], ...[TARGETING]. Since the number of clusters that contains the desirable data are then very small, so the original query Q can be evaluated by all PDA-DB's coordinated by Petri Net control. Thus, the TARGET1, ...TARGETING can be retrieved most efficiently and quickly.

Note that the data in tables [TABLE1], ...[TABLEAUX] are stored in proper secure clusters. For convenience, let us call the output of Q be LowView. When the user issues Q to retrieve a sub-relation, named LowView. Effectively, he is defining another relation LowView. The relation LowView will get its data from its proper clusters, not from HIGHVIEW. So no data are actually flowed from HighView to LowView. Such a clustering technique allows us to execute all relational operators without

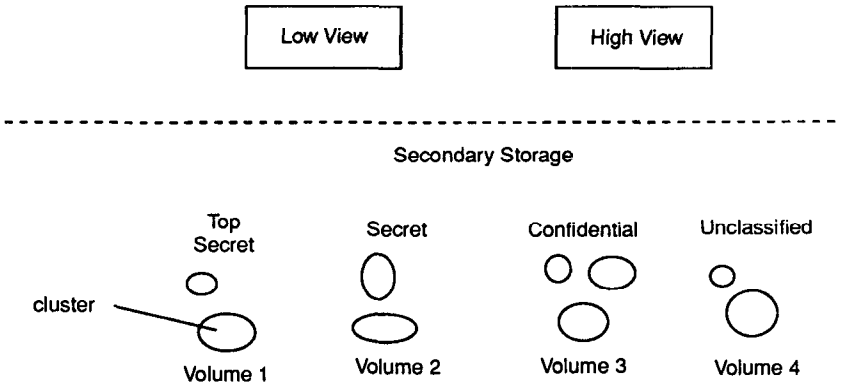

Figure 7: Secure temporal clustering

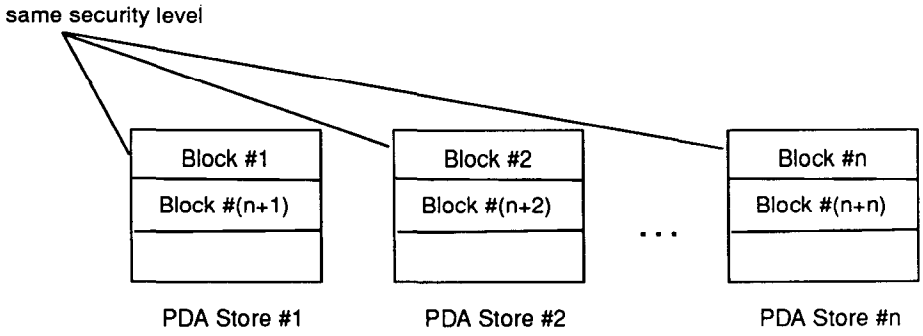

Figure 8: Another view of secure temporal clustering

using any trusted subjects, even though the data flow appears to be downward flowing (but actually not).

With clustering, all the relational operations satisfy the two properties of BLM axiom. So it is a secure operations.

7 Conclusions

On the surface, the parallel processing and security have conflicting requirements. Parallel requires declustering [DeWitt91,92], whereas security requires clustering [Lin92]. We propose temporal versions (or parallel versions) of classical concepts on storage structures. Then many classical design concepts of general purpose databases or secure databases can be ported to the parallel world. The concurrent automaton and nature computation models of hardwares [Lin91a] are used to define parallel database machines. The ABDM (Attribute Based Data model) is mapped onto the EPN-parallel machine and produce an EPN-Database machine. To support security, **secure temporal clustering equivalence** is introduced. Two data are secure clustering equivalence if it is a clustering equivalence in usual sense and their security label are the same (Section 6). An ABDM using secure clustering equivalence is called SABDM (Secure Attribute Based Data Model). SABDM is mapped onto EPN-parallel machine and we have a Secure EPN-DB machine. This illustrates that if one interprets the notion of clustering properly, a parallel machine becomes a very fast and secure database computers. The EPN-DB machines are not only secure and fast, its capability in solving harder problems (modeling) also increased in the order of magnitude. We believe this capability will surface when data processing becomes complex (knowledge processing).

References

[Brink87] J. E. Brink and R. J. Spillman, *Computer Architecture and Vax Assembly Language Programming*, Benjamin/Cummings, 1987.

[Date81,86,90]
C. Date, *Introduction to Database Management Systems*, Addison-Wesley, 1981,86,90.

Table 1: Table '1'

ATTRI1 CLASS1	ATTRI2 CLASS2	···	ATTRIs CLASSs
DATA1	DATA2	···	DATAs
LABEL1	LABEL2	···	LABELs

Table 2: Table 'i'

ATTRIn CLASSn	ATTRIm CLASSm	···	ATTRIi CLASSi
DATAn	DATAm	···	DATAi
LABELn	LABELm	···	LABELi

[DeWit91] D. DeWitt and J. Gray, Parallel Database Systems: The Future of Database Processing or a Passing Fad?, Preprint, 1991.

[DeWit92] D. DeWitt and J. Gray, "Parallel Database Systems: The Future of High Performance Database Systems," *Communications of the ACM*, 1992.

[ElNa89] R. Elmarsi and S. Navathe, *Fundementals of Database Systems*, The Benjamin/Cummings Publishing Company, 1989.

[HoUl76] J. Hopfcroft and J.Ullman, *Introduction to Automata Theory, Languages and Computations*, Addison-Wesly, 1976.

[Hsiao70] D. Hsiao and F. Farary, "A Formal System for Information Retrieval from Files," *Communications of ACM* Vol. 13, 2 (February 1970), pp. 67–73.

[Hsiao88] D. Hsiao and G. Hoppenstand, "Secure Access Control with High Access Precision: An Efficient Approach to Multilevel Security," *Database Security, II: Status and Prospects*, 1989.

[Hsiao91] D. Hsiao, "A Parallel, Scalable, and Microprocessor-Based Database Computer for Performance Gains and Capacity Grows," *IEEE Micro*, December 1991.

[Jen81] Jensen, K., Color Petri Nets and the Invariant Methods, Theoretical Computer Science, 1981.

[Lin82] Trueblood, R.,Lin, T. Y. and Kerschberg, L., Analysis of Computer Security Using Petri Net Theory, Second International Conferences in Computer Science, Santiago, Chile, 1982.

[Lin89a] T. Y. Lin, "Commutative Security Algebra and Aggregation," *Research Direction in Database Security, II, Proceedings of the Second RADC Workshop on Database Security*, December 22, 1989, pp. 167–190.

[Lin89b] T. Y. Lin, "Some Remarks On Inference Controller," *Research Direction in Database Security, II, Proceedings of the Second RADC Workshop on Database Security*, December 22, 1989, pp.158–166.

[Lin89c] T. Y. Lin, "Chinese Wall Security Policy—An Aggressive Model," *Proceedings of the Fifth Aerospace Computer Security Application Conference*, Tuscon, AZ, December 4-8, 1989, pp. 282–289.

[Lin90a] T. Y. Lin, "Extended Petri Nets and Peterson's Conjecture," *Proceedngs of Fourth Annual Parallel Symposium*, April, 1990.

[Lin90b] T. Y. Lin, "Message and Noncommutative Aggregation", *Third RADC Workshop on Database Security*, May 1991, pp. 106–116. (Coauthor: Al-Eifan. Final Revision of Message and Inference Aggregation, *Third RADC Workshop on Database Security*, June 1990.)

[Lin90c] T. Y. Lin, "Multilevel Database and Universal Security Algebra," *Third RADC Workshop on Database Security*, May 1991, pp. 96-105. (Final Revision of Aggregation and Guidelines for SSO, Third RADC Workshop on Database Security, June 1990.)

[Lin90d] T. Y. Lin, "Security Algebra and Formal Models," *Database Security, III: Status and Prospects*, edited by D. Spooner and C. E. Landwehr, North Holland, 1990, pp. 75-96. (Final revision of Security Algebra and Formal Models, Proceedings of IFIP WG11.3 Workshop on Database Security, September 5-7, 1989 (with L. Kerschberg and R. Trueblood) paper 15, with L. Kerschberg and R. Trueblood).

[Lin90e] T. Y. Lin, "Probabilistic Measure on Aggregation," *Proceeding of 6th Annual Computer Security Application Conference*, Tucson, Arizona, December 3-7, 1990, pp. 286–294.

[Lin90f] T. Y. Lin, "Multilevel Database and Aggregated Security Algebra," *Database Security, IV: Status and Prospects*, edited by S. Jajodia and C. E. Landwehr, North Holland, 1991 , pp. 325-348. (Final Revision of Database, Aggregation and Security Algebra, IFIP WG11.3 Workshop on Database Security, Halifax, UK, September 18-21, 1990.)

[Lin90g] T. Y. Lin, "Rough Sets, Neighborhood Systems and Approximation," *Fifth International Symposium on Methodologies of Intelligent Systems, Selected Papers*, Knoxville, Tennessee, October 25-27, 1990, pp. 130-141. (Coauthors: Q.Liu and K. J. Huang).

[Lin90h] T. Y. Lin, "A Model of Topological Reasoning Expert System with Application to an Expert System for Computer-Aided Diagnosis and Treatment in Acupuncture and Moxibustion," *International Symposium on Expert Systems and Neural Network Theory and Application*, Hawaii, August 15-17 1990, pp. 123-126. (Coauthors: Qing Liu and K.J. Huang)

[Lin91a] T. Y. Lin, "Concurrent automata and Parallel Architecture," *Proceeding of 1991 International Conference on Parallel Processing*, August, 1991.

[Lin91b] T. Y. Lin, "'Inference' Free Multilevel Database System," *Proceedings of the Fourth RADC Database Security Workshop*, Little Compton, RI, April, 1991.

[Lin91c] T. Y. Lin, "Entropy, Ordering and Aggregation," *Proceedings of the Fourth RADC Database Security Workshop*, Little Compton, RI, April, 1991.

[Lin91d] T. Y. Lin, *Topological and Fuzzy Rough Sets*, Kluwer Academic Publishers.

[Lin92a] T. Y. Lin, "Attibute Based Data Model and Polyinstantiation," *IFIP Congress*, September 7-12, 1992.

[Lin92b] T. Y. Lin and David Hsiao, "Rough Sets in AI as Clustering in Database," *First Workshop on Rough Set Theory*, September 2-5, 1992.

[Lin92c] T. Y. Lin and David Hsiao, "Attribute Based Data Model—A Mathematical Characterization." In preparation, 1992.

[Mura89] Tadao Murada, "Petri Nets: Properties, Analysis, and Applications," *Proceeding of IEEE* 77, 4, 1989, pp.541–580.

[Pet77] J. Peterson, "Petri nets," *ACM Computing Surveys*, 1977, pp.223–252.

[Pet81] J. Peterson, *Petri net theory and the modeling of systems*, Prentice Hall, Englewood Cliff, N.J., 1981.

[Tann87] Andrew Tannenbaum, Operating Systems: Design and Implementation, Prentice-Hall, 1987.

[Wong71] Wong, E and Chiang, T.C. "Canonical Structure in Attribute Based File Organization," *Communications of the ACM* 14, 9, 1971.