Active Rule System for Adaptive Mobile Data Access *

Shiow-yang Wu Shih-Hsun Ho

Department of Computer Science and Information Engineering
National Dong Hwa University
Hualien, Taiwan, R.O.C.
showyang@csie.ndhu.edu.tw,

Abstract. We advocate the employment of active rule systems for adaptive mobile information services. We propose a modular framework and a mobile rule processing technique with database connectivity. The technique combines static and dynamic analysis to uncover data access semantics of a rule program which facilitates intelligent caching and prefetching to conserve bandwidth, reduce processing cost, and support disconnected operations. We devise a performance model to compare our approach with traditional approach. Trace-driven simulation successfully demonstrates the feasibility and performance improvement.

1 Introduction

Mobile computing is characterized by limited resources, low bandwidth, and dynamically changing environment[1] which offers a great opportunity for applying intelligent techniques toward the provision of responsive information services. As a motivating example, imagine the scenario in which a mobile user on a moving vehicle is approaching a mountain area where radio signal is expected to be weak or completely blocked. According to the current traffic condition and also from the speed and direction of the vehicle, no mobile support station will be available for the next half an hour. For assisting the user who must finish her work for an important business meeting, the underlying information system has several choices. The easiest way is to take no action and run the risk of sudden disconnection, unfinished work, and a failed business trip. More preferably, a highly adaptive system can sense the current situation and take the initiative in preparing for disconnected operation. Our goal is to develop intelligent techniques for building responsive and effective mobile information systems. In particular, we advocate the employment of active rule system for intelligent adaptation.

Rule systems have been successfully used in providing many desirable features for traditional database systems [13]. The active rule component effectively turns a passive database into one that can react dynamically to status changes and respond actively with rule inference mechanism. For mobile data management, we

* This work was supported in part by National Science Council under project number NSC 88-2213-E-259-001.

found the active rule concept to be a powerful tool for intelligent adaptation and proposed an adaptive mobile information system framework in an earlier paper [14]. In this paper, we present the underlying rule processing technique which enables effective rule execution under various resource constraint and operating conditions. We propose a mobile data management and rule execution strategy with intelligent caching and prefetching to facilitate seamless information processing regardless of the connection status. We devise a performance model to compare our approach with traditional on-demand approach. Trace-driven simulation results successfully demonstrate the feasibility of our approach and the potential for significant performance improvement.

The rest of the paper is organized as follows. Section 2 provides a survey of related work. Section 3 presents our framework for mobile rule processing. In Section 4, we introduce the intelligent caching and prefetching technique. A performance model is introduced in Section 5 to analyze the effectiveness of the proposed techniques. In Section 6, we discuss a trace-driven simulation method and the results of various experiments. Section 7 concludes the paper.

2 Related Work

The need for intelligent adaptation is considered essential for mobile data management [5]. Similar ideas have been discussed under terms like context-aware [11], application-aware [10], environment-directed [12], and adaptive [3] information systems. The decision on when and how to adapt can be made by either the underlying system (*application-transparent adaptation*) or the application programs (*application-aware adaptation*) [9]. Application-transparent implies that no change is needed to existing applications which also means no application-specific feature can be exploited. On the other hand, the performance and flexibility offered by the application-aware adaptation may very well come with higher complexity and software development cost. Our framework integrates existing rule system and database systems for intelligent adaptation which offers the benefits of both application-transparent and application-aware adaptation.

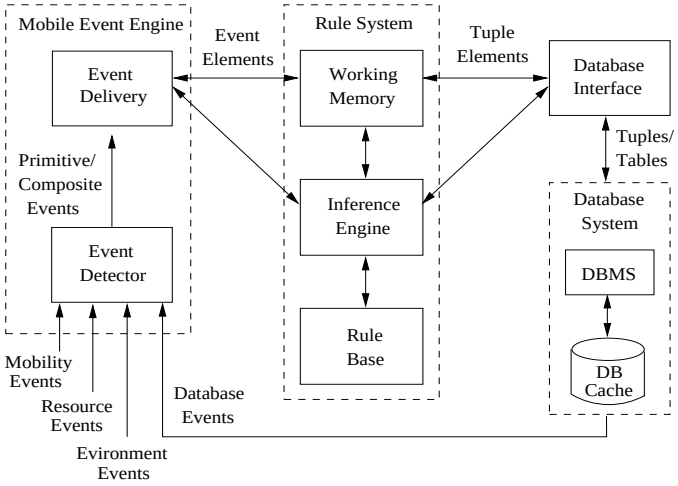
A database system with an integrated rule system is called an *active database system*[13]. A rule system is composed of a *working memory*, a set of *rules*, and an *inference engine*. Working memory is a global space composed of data objects called *working memory elements* (WMEs) representing the system state. A rule is a condition-action pair. The inference engine provides a three-phase cyclic execution model of *condition evaluation*, *conflict-resolution* and *action firing*. A rule with a set of WMEs satisfying the conditions is called an *instantiation*. The set of all instantiations constitutes the *conflict set*. Conflict-resolution selects one instantiation from the conflict set for firing. Firing an instantiation executes the actions which may add, delete, or modify WMEs in the working memory. The cycle repeats until no more rule can be fired. Rules in active database systems are normally enhanced with event triggering capability and known as *event-condition-action rules* (ECA rules)[2, 7]. An ECA rule is triggered on the occurrence of certain events. The condition part of the rule is evaluated against

the current state to determine whether the conditions are satisfied. The action part of the rule is then executed whenever the conditions are matched.

ECA rules are natural candidate for adaptive mobile information access. Resource and environment changes can be modeled as events. The condition evaluation, action firing, and rule inference mechanisms can be used for making proper adaptation decision. Our main contribution is to provide a framework and the underlying rule processing techniques for building mobile information systems that can respond actively to resource and environment changes.

3 A Framework for Adaptive Mobile Information Access

We describe our framework in brief. Readers are referred to [14] for more information. As depicted in Figure 1, we proposed the integration of a *mobile event engine*, a *rule system*, and a *database system* for adaptive information access. The mobile event engine is to detect status changes and to trigger the rule system. In additional to the database and temporal events [13], we suggest the inclusion of mobility, resource, and environment events. Whenever the status of a mobility variable (location, speed, etc.), a resource or an environment variable has changed to a critical level, a corresponding event occurs. Such changes are detected by the *event detector* and treated as *primitive events* which can be combined to form *composite events*. The *event delivery module* transforms the events into *event elements* which are in the same format as WMEs of the rule system. In this way, the event engine can be easily integrated with the rule system.



F 1. odular Framework for Building daptive obile Information Systems.

The *database interface* integrates rule and database systems. Tuples from the database are converted into *tuple elements* for the rule system. ata generated

within the rule system can be saved into database. y conforming to standard, the interface can interact with multiple and heterogeneous databases.

In this paper, we focus on the mobile rule processing technique which is designed based on the following observations

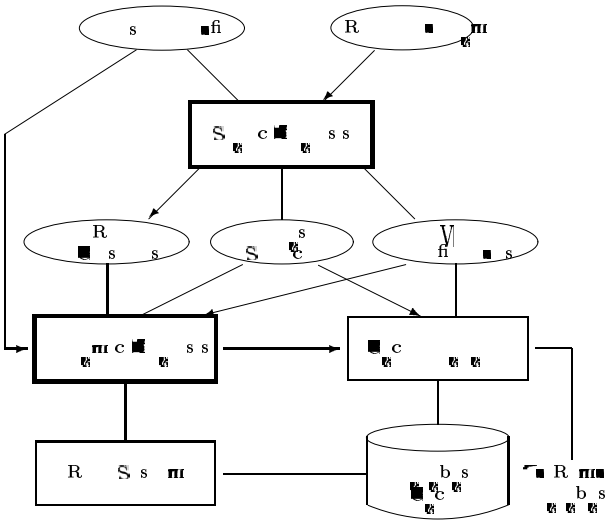
> Most rule programs execute in phases. uring each phase, only a small subset of the rules are active. These rules exhibit strong data access locality that can be determined by examining the condition part of the rules.
> It is common to use a *goal element* to control phase execution. All rules of the same phase match the same goal element. *ase c anging rules* are used to update the goal elements for switching between phases.
> The execution of a rule results in small changes to the database. An active rule which is not selected for firing in the current cycle is likely to remain valid in the next cycle since most data remain unchanged. Therefore, it is beneficial to keep the data available through caching.

Our framework is organized as an integration of static and dynamic analysis for prefetching and caching as depicted in Figure 2. The phase execution behavior and strong locality of data access suggest a strategy that prefetch and cache data according to the phase execution semantics. Static analysis can uncover the phase structure and partition the rules into clusters. The purpose is to group together rules with similar data access pattern. The data accessed by the rules in a cluster can be determined by analyzing the condition part of these rules. The result of the static analysis is the rule clusters and a set of *view definitions* corresponding to the data accessed by each cluster. These information are used by the cache manager to determine when and what data to prefetch and retain in the cache.
ynamic analysis is to monitor rule execution and to adjust prefetching and caching policy if necessary. User profiles are used whenever applicable to improve the effectiveness of the prefetching and caching decisions. etails of our technique will be presented in the next section.

redictive Cac in and refetc in

hase structure analysis described above can be done by grouping together all rules that match the same goal element or by using *data dependency grap* [4] and *clustering* techniques [6]. The set of view definitions extracted for a rule cluster is formulated to include all data that can possibly be accessed by any rule in the cluster. User profile is applied to limit the scope and range.

We apply the technique on the following rules. Continuing with the example in Section 1, the mobile user is about to report her reorganization plan to the CEO at the headquarter. For an unexpected last minute change, part of the proposal must be modified along the way. Reorganization planning strategy is determined based on bandwidth. In normal condition, a detail planning can be taken. In case that the bandwidth falls and the time remaining is short, a simple salary cut strategy is adopted. We also assume that the manager has an authority level to examine employee records with salary lower than 100,000 dollars.



F g . Integration of Static and Dynamic nalysis for obile Rule rocessing.

```
r\ e ¯tr te y_¯e e■ti
   A p  se ■    i   r\ e t  se e■t   simp e str te y if t e
    v i  b e b  dwidt  is   w   d t e time rem i i   is s  rt.
   (b  dwidt  < 1 Mbps) AND (time_rem i  < 30 mi \tes)
if ( :G   )            e eme t exist
t e
    .■\rre t_     = ADJU¯T_¯ALARY }

r\ e ¯   ry_C\t
   C\t 10%  ff t e s   ry  f  i  -p id emp  yee (s   ry > 80000)
   f r dep rtme t wit   e  tive e r i  s.
if ( :G   ::  .■\rre t_     == ADJU¯T_¯ALARY)
   (d:Dep rtme t:: d.e r i  s < 0)
   (e:Emp  yee:: e.dep rtme t == d.  me & e.s   ry > 80000)
t e
   e.s   ry = e.s   ry - e.s   ry 10 }

r\ e ¯   ry_R ise
   R ise 10% t e s   ry  f   w-p id emp  yee (s   ry < 10000)
   f r dep rtme t wit  e r i  s >   e mi  i  .
if ( :G   ::  .■\rre t_     == ADJU¯T_¯ALARY)
   (d:Dep rtme t:: d.e r i  s > 1000000)
   (e:Emp  yee:: e.dep rtme t == d.  me & e.s   ry < 10000)
t e
   e.s   ry = e.s   ry + e.s   ry 10 }
```

uring the phase structure analysis, all ADJU¯T_¯ALARY rules will be grouped together into the same partition. The data requirement can be extracted from the test conditions and represented by the following view definitions.

```
¯ELECT *
FROM   Dep rtme t
WHERE  e r i  s < 0 OR e r i  s > 1000000

¯ELECT *
FROM   Emp  yee
WHERE  s   ry > 80000 OR s   ry < 10000
```

A closer look at the rules reveals the disjointness of the test conditions which separates the rules into two clusters. The view definitions becomes

```
¯ELECT *
FROM   Dep rtme t
WHERE  e r i  s < 0

¯ELECT *
FROM   Emp  yee
WHERE  s   ry > 80000
```

for the first cluster consisting of the ¯ ry_C\t rule, and similarly for the second cluster consisting of the ¯ ry_R ise rule. The range of data can be further constrained by considering the manager's authority level

```
¯ELECT *
FROM   Emp  yee
WHERE  s   ry > 80000 AND s   ry < 100000
```

Only the data in the views are needed for the rule cluster to execute properly which significantly reduces the volume of data transfer and conserves valuable bandwidth. We now describe our mobile rule execution strategy.

Apply static analysis to partition the rules and extract view definitions.
Upon execution of a phase, prefetch all views of that phase in one connection.
Log all updates. At the end of a phase, send the net updates back to the remote database in one connection.
Upon disconnection, simply use the data in the cache to proceed normally.
As long as all views have been cached, the execution remains valid.

Intuitively, our approach improves latency by keeping currently used and to be used data nearby. andwidth is conserved by loading only the necessary views. isconnected operations are supported by prefetching and caching data for the current and subsequent phases. In later sections we will demonstrate that this simple approach can result in significant saving in rule execution cost.

5 erformance Model and Comparison

We devise a performance model and compare our approach with traditional approach which accesses data on-demand. We assume that each request results in a single unit of data access. The model is described by the following parameters

C_c	Cost for setting up a connection.
C_t	Unit cost of data transmission.
C_{io}	Unit I/O cost.
R_p	Cache hit ratio (with prefetching).
R_{np}	Cache hit ratio (w/o prefetching).
T	Total number of data requests.
V_p	Total units of prefetched data.
H	The additional hits results from the prefetching of V_p (i.e. $T*(R_p-R_{np})$.

A cache hit can be satisfied locally. A cache miss incurs the cost of connection setup, remote access, and data transfer. The total cost for the approach with and without prefetching (E_p and E_{np}) can be modeled as

$$E_{np} = T * R_{np} * C_{io} + T * (1 - R_{np}) * (C_c + 2C_{io} + C_t)$$
$$E_p = (C_c + (2C_{io} + C_t) * V_p) + T * R_p * C_{io} + T * (1 - R_p) * (C_c + 2C_{io} + C_t)$$

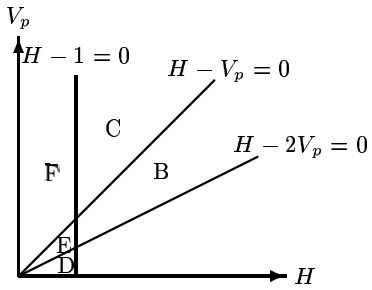
where the first item of E_p is the cost of prefetching. For the new approach to be effective, we must have $E_{np} - E_p > 0$. y simple algebraic manipulation, we obtain the relationship between E_{np} and E_p as follows.

$$E_{np} - E_p = (H - 1)C_c + (H - 2V_p)C_{io} + (H - V_p)C_t$$

This can be depicted in Figure 3 which shows the desired level of accuracy in prefetching. Region A ($H > 2V_p$) represents the case when prefetching will definitely have an advantage. Region ($2V_p > H > V_p$) stands for the situation that prefetching does not guarantee but highly likely to provide better performance. This is because $C_t \gg C_{io}$ in mobile environment. If C_{io} can be safely ignored, then prefetching is beneficial in both region A and . It is not cost worthy in region C, which means when the additional hits is smaller than the prefetched units ($H < V_p$), the overhead prevails. Regions , E, and F are cases when none of the data prefetched is actually used which are very unlikely.

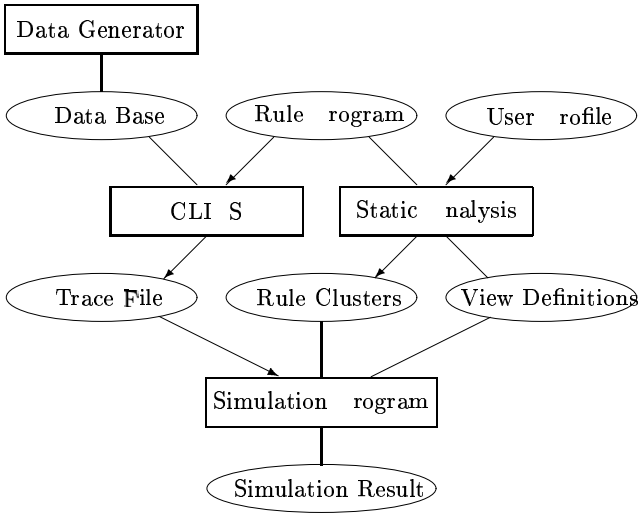
6 erformance Evaluation

We have devised a trace-driven simulation method as depicted in Figure 4. The CLI S rule system [8] was used to generate the execution trace. The test programs are scalable rule programs that access databases of arbitrary size generated by a data generator. The trace files are produced by running the rule programs on CLI S over databases of increasing size. Each trace file contains the detail record of every single rule firing and the data accessed by that rule.



F **3.** Effectiveness of prefetching vs. no prefetching.

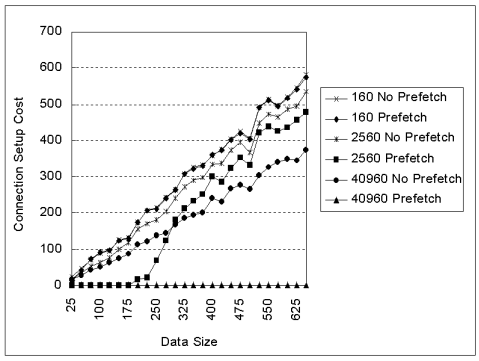
Static analysis is applied to produce the rule clusters and view definitions. The simulation program takes a trace file, the rule clusters and view definitions as input and produces detail figures of the execution cost under various settings. Given a set of parameters such as the cache size, C_c, C_t, C_{io}, and the replacement policy, the simulation program parses the trace file and performs a pseudo execution of the rule program to accumulate all relevant costs. We are interested in connection setup, I/O, and transmission cost. The connection cost is important since each connection consumes valuable power and bandwidth. I/O and transmission cost reduction improves response time and saves energy as well.



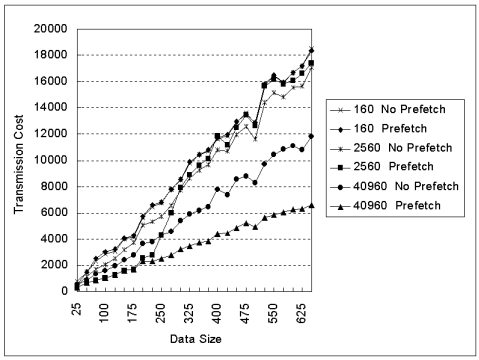
F **4.** Simulation ethod.

Figure 5, 6, and 7 present the comparison between on-demand approach and our scheme with small(160), median(2560), and large(40960) disk cache units. Our approach is superior in all three dimensions. Take Figure 5 as an example,

our approach incurs much lower connection cost. This is because we need only one connection to prefetch all data required in a particular phase. When the cache is large enough(to hold the largest view), the number of connections required is exactly the number of phase changes during the program execution.



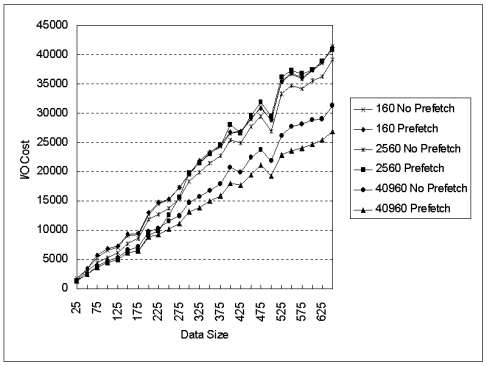
F **5.** Connection cost comparison.



F **6.** Transmission cost comparison.

7 Conclusions and Future Work

We presented a framework and techniques for building adaptive mobile information systems. The techniques employs semantic analysis to predict future data requirement. Significant cost saving can be achieved if the data is prefetched and cached at the right time. Locally cached data also facilitate disconnected operations. To complete the design, a mobile event engine is necessary. We are working on the event subsystem as well as its integration with other modules.



F **7.** I/O cost comparison.

References

1. G. Forman and J. Zahorjan. The challenges of mobile computing. *IEEE Computer*, 27(6):38–47, pr. 1994.
2. E. N. Hanson and J. Widom. n overview of production rules in database systems. *The Knowledge Engineering Review*, 8(2):121–143, June 1993.
3. T. Imielinski and S. Vishwanathan. daptive wireless information systems. Technical report, Department of Computer Science, Rutgers University, 1994.
4. T. Ishida and S. J. Stolfo. Toward the parallel execution of rules in production system programs. In *IEEE Intl. Conf. on Parallel Processing*, pages 568–574, 1985.
5. R. H. Katz. daptation and mobility in wireless information systems. *IEEE Personal Communications*, 1(1):6–17, 1994.
6. D. . iranker, C. Kuo, and J. Browne. arallelizing compilation of rule-based programs. In *IEEE Intl. Conf. on Parallel Processing, Vol. II*, pages 247–251, 1990.
7. N. W. aton, editor. *Active Rules in Database Systems*. Springer, New York, 1999.
8. G. Riley. CLI S: tool for building expert systems. (`http://www.jsc.nasa.gov/~clips/CLIPS.html`).
9. . Satyanarayanan. obile information access. *IEEE Personal Communications*, 3(1), Feb. 1996.
10. . Satyanarayanan, B. Noble, . Kumar, and . rice. pplication-aware adaptation for mobile computing. *Operating System Review*, 29, Jan. 1995.
11. B. Schilit, N. dams, and R. Want. Context-aware mobile applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, C , U.S., Dec. 1994.
12. G. Welling and B. R. Badrinath. objects: rogramming support for environment directed application policies in mobile computing. In *ECOOP'95 Workshop on Mobility and Replication*, ug. 1995.
13. J. Widom and S. Ceri. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. organ Kaufmann, San Francisco, California, 1996.
14. S. Wu and C.-S. Chang. n active database framework for adaptive mobile data access. In *Workshop on Mobile Data Access, 17th International Conference on Conceptual Modeling, Singapore*, 1998.