Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 34 (2014) 330 – 337
Procedia Computer Science

The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC-2014)

Access-Pattern Aware Checkpointing Data Storage Scheme for Mobile Computing Environment

Xiang Li[a,b], Mei Yang[b], ChaoGuang Men[a], YingTao Jiang[b], Kalum Udagepola[c] *

[a]*College of Computer Science and Technology, Harbin Engineering University,145 NanTong Street, NanGang District, Harbin, China*
[b]*Department of Electrical and Computer Engineering, University of Nevada Las Vegas, 4505S. Maryland Parkway, Las Vegas, 89154, USA*
[c]*Faculty of Computing and Information Technology, North Branch, King Abdulaziz University, Jeddah, Saudi Arabia*

Abstract

The rapid development of communication technology and mobile devices has facilitated the mobile computing applications such as location-based services, and information sharing. Characteristics of the mobile computing system, such as the space limitation, mobility handling, low bandwidth and limited battery life, make mobile computing applications more prone to failures. Checkpoint and rollback recovery technology, as a fault tolerance method for continuing services in mobile computing environment, is researched in this paper. Based on user access patterns, mainly considering the visit time of the mobile host(MH) to the sojourn mobile support stations (MSSs), a checkpointing data storage scheme is proposed. Only if the stay time in the current mobile support station is long enough, the mobile host should do the checkpoint and store the checkpoint data on the sojourn mobile support station. Based on the visited time to the MSS, the scheme manages checkpointing data high efficiently which is more cognitively flexible and adapt to the nomadic and mobile computing environmental conditions.

Selection and peer-review under responsibility of Conference Program Chairs
Keywords: Mobile Computing; Checkpoint and Rollback Rocovery; Access Pattern; Mobile Host; Mobile Support Station

1. Introduction

Rapid development of communication technology from wired to wireless network led almost all service oriented systems to “all time everywhere” service from “anywhere anytime” service[1]. This mobile computing enables users to access and exchange information conveniently and seamlessly while they roam around in mobile environments, and

* Corresponding author. Tel.: +1-702-423-4974.
E-mail address: leexiang@hrbeu.edu.cn

Selection and peer-review under responsibility of Conference Program Chairs
doi:10.1016/j.procs.2014.07.032

to work continuously everywhere. Thus, mobile computing requires techniques to provide fault tolerance for continuing services despite failures. Checkpoint and rollback recovery is a popular technique for fault tolerance in mobile computing.

Mobile computing environments pose challenging problems in designing checkpoint and rollback recovery strategy because of its special properties, such as space limitation, mobility handling, low bandwidth, limited battery life[2,3]. Considering these properties, checkpointing data storage management plays a very important role for checkpoint and rollback recovery. Effectively storing and managing checkpoint data, implementing the fast access and efficient utilization of checkpoint data, are key factors to enhance checkpointing performance in mobile computing since mobile computing systems are more prone to failures[4].

It has been known through various surveys that the mobile users' behaviors exhibit various degrees of regularity[5]. Actually, the majority of users do not travel at random in their daily life. They navigate from place to place with specific purposes in mind. In many cases, the location movement and service invocation of mobile users have regular patterns. Using such mobility and service patterns, it is potentially beneficial to facilitate network and data management. In this paper, based on user access pattern, a checkpointing data storage scheme is proposed for specified application in mobile computing environment. If the stay time in a MSS exceeds a threshold the MH_i should do the checkpoint and store the checkpoint data on the sojourn mobile support station. Since the sojourn time is long enough it is possible to do the checkpoint high efficiently.

2. Related Works

Some distributed storage management schemes of checkpointing data has been proposed. For fast recovery, it is desirable the checkpoints and message logs to be near the MH on recovery, and hence, checkpoints and logs of a MH in [6] keep moving as the MH performs the handoff between cells. Two schemes for state-saving, No Logging and Logging, are proposed. And three ways to transfer the recovery information are given, namely, Pessimistic (Pessimistic scheme is always known as Eager scheme), Lazy, and Trickle strategy[6]. One suggestion made in [7] utilizes the home of each MH to maintain the recovery information. As a MH moves, it transfers checkpoints or logs to the home, and in case of a failure, it can find the recovery information at home. However, if the MH is far from home, the transfer cost can also be a problem. In [8] a checkpoint protocol is proposed which achieves global consistent checkpoint without additional messaging. The protocol saves soft checkpoints locally in the mobile host, and stores hard checkpoints in stable storage. In [9], a compromising checkpointing data storage strategy is proposed in which high-priority recovery information will be transferred to the cell during handoff and other low-priority recovery information will be transferred while failure recovery.

In [10], two movement-based schemes, namely, frequency-based and distance-based scheme, are proposed. In these two schemes the latest checkpoint and message logs are transferred to the now residing MSS while the number of handoffs in the frequency-based scheme exceeds k, and the distance between the MSS in which MH_i is now residing and the MSS carrying the latest checkpoint of MH_i exceeds a threshold TH_i in distance-based scheme. Similarity, movement-based checkpointing scheme in [11] takes a checkpoint only after a threshold of the number of mobility handoffs has been exceeded. Its optimal threshold is governed by the failure rate, log arrival rate, and the mobility rate of the application and the mobile host.

Region-based storage management schemes proposed in [12] assign a recovery manager for a group of cells taking care of the recovery for the mobile hosts within the region. The scheme can reduce the recovery cost by transferring partial or whole recovery information within a region; or it can reduce the hand-off cost by restricting the transfer. The scheme synthetically utilizes the method of carrying recovery information by MH while moving proposed in [6] and fast recovery scheme of centralized storing information by MH proposed in [7]. A region-based distributed storage management scheme for mobile environments was presented in [13]. Here, the patterns of MHs' movement are categorized into three steps: moving within a region, moving around neighbor regions, moving across regions. When an MH moves within a region, the recovery information is not migrated. When the MH moves around the neighbor regions, each RM (*recovery manager*, a MSS in the region, which takes the role of the home for the MHs in the region) collects the information for fast recovery. Only when the MH moves far from the previous region, the recovery information is migrated to the current region manager.

An adaptive protocol integrating leasing mechanism proposed in [2] manages storage for base stations. Each process negotiates with the storage manager to determine the size and duration of the lease. The adaptive checkpointing with leasing dynamically determines the appropriate location to store checkpoints based on available resources, controls disk-space effectively and reduces checkpointing overhead.

So far there are less checkpoint data storage management strategies which are specifically designed for mobile computing environments and most are based on the structure of the mobile support station of mobile computing systems. For each of the mobile computing models, taking different checkpoint data storage management strategy, message logs needed to be recorded will be different, and so different the costs of the checkpointing will have. Based on the visit history of a moving host/user, the current sojourn time on the new mobile support station was taken as the only factor of the user access pattern to transfer the checkpoints and logs to the new MSS or not in this paper. In this work, the failure-free and failure costs were widely dispersed and varied by the access pattern, which showed good flexibility and adaptability to the mobile application environment.

3. Mobile Computing System

The system consists of a set of mobile hosts who are free to move around. At any time, they maintain network connectivity through a wireless link to a static MSS. The MSSs are interconnected through high speed static wired networks. A MSS handles all communications to and from MHs within its area of influence known as a cell. The delivering and receiving messages of MH in the cell are processing by MSS. The wired and wireless network protocols are assumed to provide reliable FIFO delivery of messages with arbitrary delay to the application.

A mobile computing system MCS= 〈N, C〉 is composed of a set of nodes N and a set of channels C. The set of nodes N=M∪S can be divided into two types, M={ MH_1, MH_2,..., MH_m } is the set of MHs, and S={MSS_1, MSS_2,..., MSS_s} is the set of static nodes acting as MSSs with extra processing power, communication and storage capabilities. Because of mobility, an active MH can move freely from one cell to another, and the local MSS responsible for the MH is changed correspondingly. This process, known as a handoff, is transparent to the MH.

As a MH_i moves from one cell to another cell, the message logs of MH_i may become distributed over the storages of a number of MSS_s. And also, MH_i may be far away from the MSS which carries the latest checkpoint when a failure occurs. A mobile host, MH_i, after a failure, must have the ability to locate its latest checkpoint and logged messages for the following consistent recovery.

In the checkpoint and rollback recovery protocols, the MH periodically backups the state of the local process, resulting in new checkpoint information stored at the current MSS. Between two checkpoint events, each write-event is also logged at the storage of MSSs. When the MH fails, and subsequently recovers, the MH reads this persistent information stored on MSSs to roll back to a state saved at the corresponding checkpoint, and then re-executes write-events through reading log entries saved after the last checkpoint.

4. Access-Pattern Aware Checkpointing Data Storage

4.1 The Data Structure and Denotations

To ensure the consistency of the rollback recovery after a fault, the failure MH needs to locate and retrieve its required checkpoint and log messages. For better management of such recovery information during the handoff process, the local MSS maintains a tuple $Trace_i$ for each MH_i in its cell.

$Trace_i$ contains variables, such as, *cp_seq*, *cp_loc*, *log_set*, *cp_period*, *fail_rate*, *mov_rate*, *vis_rate*. Exactly, *cp_seq* denotes the sequence number of the latest checkpoint and *cp_loc* denotes the identifier of the MSS carrying the latest checkpoint of MH_i; *log_set* is a list established for the local connected MSS to save the identifiers of MSSs which hold determinants logged after the latest checkpoint; *cp_period* denotes the fixed interval consecutive checkpoints of MH_i; *fail_rate* and *mov_rate* denote the failure rate and handoff rate of the process P_i at MH_i respectively. Especially, *vis_rate*, indicated by θ_i^p, is the rate at which MH_i visits the mobile support station MSS_p, and it is decided by the visit frequency, time, or other factors. The $Trace_i$ of MH_i is used to manage the handoff process, locate and retrieve the required information during recovery processes.

4.2 Checkpointing and Logging

A node periodically takes a checkpoint. When the checkpoint is successfully being done by a mobile host MH_i, it will first update its $Trace_i$. MH_i then sends the checkpoint with $Trace_i$ to its current MSS. On the receipt of the checkpoint, MSS saves the checkpoint and the related information into the stable storage.

Let M_i^{α} denotes the α-th message delivered to MH_i, and each message, M_i^{α} is identified by the pair of integers, (i, α). With the application messages from MSS_p to the MHs in the cell, MSS_p also logs the messages related to the mobility, such as the ***join***, ***leav*e**, ***disconnect***, ***reconnect***, **and** ***sleep*** messages. Any of these messages sent from a mobile host, MH_i, must carry the value of $m_i^{rcv\_seq}$, to count the number of messages MH_i has received. The logs of these messages are used to trace the movement of each MH_i during the recovery.

The message logging operation is depicted in Fig.1.

> **When** MSS_p delivers a message, M_i^{α}, to MH_i:
> Save M_i^{α} into Stable_Log_Space;
> If ($p \notin Trace_i.\log\_set$) $Trace_i.\log\_set = Trace_i.\log\_set \cup \{p\}$;
> Delivers M_i^{α} ; $\alpha = \alpha + 1$; Save [(i,α)] into Stable_Log_Space;
> **When** MH_i enters into a Cell managed by MSS_p;
> MSS_p saves [($Trace_i$; $m_i^{rcv\_seq}$)] into Stable_Storage;
> **When** MSS_p receives a message, M, from MH_i:
> If ($M \in \{join, leave, disconnect, reconnect, sleep\}$) Saves [M;($m_i^{rcv\_seq}$)] into Stable_Log_Space;

Fig.1. Message Logging

4.3 Distributed Storage Management

While MH_i moves from the cell of MSS_p to another cell of MSS_q, MH_i first closes its connectivity to the MSS_p through sending the ***disconnect*** request and receiving the reply. After receiving message ***join*** from the MH_i, the new local MSS_q invokes the handoff operation. Supposed that V is the threshold of *vis_rate*, i.e. the rate after which the MH transfers the checkpoint and log to the new cell's mobile support station MSS_p during handoff. If $\theta_i^p > V$ hold, then MH_i implements *Pessimisti*c handoff operation, otherwise MH_i adopts *Lazy* handoff scheme. The pseudo-code of the handoff procedure is described in Fig.2.

> **When** MSS_q receives the ***join*** message from MH_i, which moved from the cell of MSS_p
> retrieve $Trace_i$ from the previous MSS_p;
> if ($\theta_i^p > V$) /* Pessimistic_Handoff_Implement (i, $Trace_i$); */
> { Send [*recovery_information_collection*;$Trace_i.cp\_seq$;i] to $Trace_i.cp\_loc$;
> For (each $MSS_r \in Trace_i.log\_set$) Send [*recovery_information_collection*;i] to MSS_r; }
> Else /* Lazy_Handoff_Implement (i, $Trace_i$); */ Save [($Trace_i$; $m_i^{rcv\_seq}$)] into Stable_Storage;
> **When** MSS_p receives (*recovery_information_collection*; $Trace_i.cp\_seq$; i) from MSS_q;
> Send $C_i^{Trace_i.cp\_seq}$ to MSS_q; /* $C_i^{Trace_i.cp\_seq}$ denotes the $Trace_i.cp\_seq$-th checkpoint of MH_i, */
> For ($M_i^{\alpha} \in Log\_Space$) if ($\alpha > C_i^{Trace_i.cp\_seq}.m_i^{rcv\_seq}$) send M_i^{α} to MSS_q; /* *Log_Space* is the storage space of all logs */
> **When** MSS_r receives [*recovery_information_collection*; i] from MSS_q;
> Send [$M_i^{\alpha} \in Log\_Space$] to MSS_q;
> **When** MSS_q collects [$C_i^{Trace_i.cp\_seq}$; $M_i^{\alpha}s$];
> Save $C_i^{Trace_i.cp\_seq}$ into Stable_Checkpoint_Space;
> For ($\alpha = C_i^{Trace_i.cp\_seq}.m_i^{rcv\_seq} + 1;; \alpha{+}{+}$) Save M_i^{α} into Stable_Log_Space;
> $Trace_i.cp\_seq_i$=cp_seq_i; $Trace_i.cp\_loc$=q; $Trace_i.log\_set$=$\{q\}$;

Fig.2. The Handoff Management

In the procedure of *Lazy* handoff, the new local MSS_q only need to pass the related $Trace_i$ to logging mechanism for stable logging. In the procedure of *Eager* handoff, the requests are sent to collect the related recovery information at MH_i according to $Trace_i$. Upon receipt of these requests, the corresponding requested MSS_s reply with the required checkpoint and logs of MH_i saved on their storages. When have received all the related recovery information of MH_i, the new local MSS_q will complete the entire handoff processing through saving collected checkpoint and logs information *of* MH_i's into stable storage, and the related $Trace_i$ will be updated.

5. Performance Analysis

The following parameters or notations used and their descriptions are listed in table 1.

Table 1. Parameters and its Description

Parameters	Description
λ	*fail_rate*, is the MH failure rate, i.e. the rate at which the MH fail
μ_i	*mov_rate*, is the MH mobility rate, i.e. the rate at which the MH crosses cell boundaries
γ_i^p	the handoff rate of the MH to a cell, i.e. the rate at which the MH_i handoffs from other cells to the cell supported by $\boldsymbol{MSS_p}$
Nl_C	the number of messages of a checkpoint, i.e. the size of the checkpoint data, and it is designed as a constant unit
Nl_H	the number of messages logged for one handoff, and it is designed as a constant unit
VT_i^p	the visit time of the MH_i to $\boldsymbol{MSS_p}$, or the sojourn time of the MH_i in the cell supported by the $\boldsymbol{MSS_p}$

The performance of our proposed scheme, namely AccessPattern-based scheme, is compared with the following three schemes: Pessimistic/Eager and Lazy scheme in [7], and frequency-based Scheme in [11].

5.1 Failure-Free Overhead

The total checkpoint and log message size needed to be collected while handing off is evaluated for different failure-free checkpointing and recovery schemes.

1. Cost of Eager Scheme

The mean time to failure (MTTF) is T_{MTTF} (=1/λ). The MH_i crosses cell boundaries with mobility rate μ_i, the number of handoffs Nh_{MTTF} between two consecutive failures is

$$Nh_{MTTF} = T_{MTTF} \times \mu_i = \mu_i / \lambda \tag{1}$$

With eager strategy, the recovery information (the number of checkpoint and message logs) needed to transfer during failure-free, that is, the failure-free cost of the MH during two consecutive failures is

$$Cost_{ff}^{eager} = (NI_C + NI_H) \times Nh_{MTTF} = (NI_C + NI_H) \times (\mu_i \div \lambda) \tag{2}$$

2. Cost of Lazy Scheme

The checkpoint and message logs are not transferred and collected only in case of failure recovery. So the failure-free cost is zero, that is $Cost_{ff}^{lazy} = 0$.

3. Cost of Frequency-based Scheme

$$Cost_{ff}^{frequency} = (NI_C + NI_H) \times \left|\frac{Nh_{MTTF}}{k}\right| = (NI_C + NI_H) \times \left|\frac{\mu_i}{\lambda k}\right| \tag{3}$$

4. Cost of AccessPattern-based Scheme

Supposed that user access patterns of a MH_i are known as sequences pairs (MSS_r, VT_i^r), where MSS_r is the mobile support station that MH_i has visited. V is simply formulated as follows:

$$\theta_i^p = \frac{VT_i^p}{\sum VT_i^r} \tag{4}$$

With our logging strategy, the MH_i moves from a cell to another cell with visit time VT_i^p at handoff rate γ_i^p. Only if $\theta_i^p > V$, the MH_i transfers all the recovery information to the new cell's base station. The number of handoffs in which the latest checkpoint and message logs needed to collect during two consecutive failures is

$$n_{handoff} = \sum_{\theta_i^p > V} Nh_{MTTF} \times \gamma_i^p \tag{5}$$

The failure-free cost of the MH_i during two consecutive failures is

$$Cost_{ff}^{AP} = (Nl_C + Nl_H) \times n_{handoff} = (Nl_C + Nl_H) \times \sum_{\theta_i^p > V} Nh_{MTTF} \times \gamma_i^p \tag{6}$$

5.2 Failure-Recovery Overhead/Cost

The longer the mobile host stays in a MSS the higher probability of collecting recovery information will be companied. Hence, the probability of collecting recovery information while staying in a MSS is measured using the *vis_rate* directly. Also, the longer the mobile host stays in a station, the more checkpointing data are produced and the higher failure probability will be. The failure recovery costs while staying in the MSS_s are evaluated using the probability models of collecting recovery information from non-local *MSS* as follows.

1. Cost of Eager Scheme

For eager scheme, when MH handoffs from one MSS to another MSS the recovery information will be transferred to the new MSS at the same time. Hence it does not need to obtain the data for recovery while failure occurring. Subsequently, the failure-recovery cost is zero here.

2. Cost of Lazy Scheme

For lazy scheme, checkpoint and log information will not be collected until the failure occurs. The probability of visiting the MSS_p is V. And, the probability of collecting recovery information is also. For each MSS_p, the probability of collecting recovery information locally is $V * V (= (V)^2)$. Hence, the probability of collecting recovery information from non-local *MSS* is $V - (V)^2$. There are totally s mobile support stations from $\boldsymbol{MSS_1}$ to MSS_s. We get the probability of collecting recovery information non-locally as

$$P_f^{lazy} = \sum_{p=1}^{s} [\theta_i^p - (\theta_i^p)^2] = \sum_{p=1}^{s} \theta_i^p - \sum_{p=1}^{s} (\theta_i^p)^2 = 1 - \sum_{p=1}^{s} (\theta_i^p)^2 \tag{7}$$

3. Cost of Frequency-based Scheme

The MSS performs the collection of the latest checkpoint and message logs after k handoffs. For each MSS the probability of collecting recovery information non-locally is $(C_k^{k-1})(V - (V)^2)$, hence

$$P_f^{frequency} = \sum_{p=1}^{s} [C_k^{k-1} \times (\theta_i^p - (\theta_i^p)^2)] = \frac{k-1}{k} \times \left[1 - \sum_{p=1}^{s} \left(\theta_i^p\right)^2\right] \tag{8}$$

4. Cost of AccessPattern-based Scheme

For AccessPattern-based scheme, if $\theta_i^p > V$, the MH_i transfers recovery information to the new station when handoff happens. Hence, the probability of collecting recovery information non-locally while failure happening is

$$P_f^{AP} = \sum_{\forall_{MSS_p \in S}, \theta_i^p < V} (\theta_i^p - (\theta_i^p)^2) \tag{9}$$

5.3 Simulation Results and Discussion

Using MATLAB simulator, the mobile host moves between 20 mobile support stations. The following parameters were kept constant across all the runs: Nl_c=40, Nl_H=1. To obtain the performance, the following variables were used: $\lambda \in [0.00001, 0.0001]$, $\mu_i \in [0.001, 0.01]$. Four schemes are: Eager scheme denoted by EL, Lazy scheme denoted by LL, Frequency-based scheme with k values 3,5,7, denoted by 3-F, 5-F, 7-F,.etc, respectively, and AccessPattern-based scheme with V values of 0.08, denoted by 0.08-V, ...etc.

Fig.3 shows the failure-free cost of our proposed AccessPattern-based scheme using equation (6). When V varies from 0.015 to 0.15, with the increase of the threshold V the overhead produces dispersed distribution. The failure free cost of our scheme will not fully decrease with V, but normally the failure free cost will decrease with the increase of V. This is because in our scheme we take sojourn time as the main factor to collect the checkpoint recovery information or not. Only after the sojourn time is long enough and exceeds the threshold value V the checkpoint recovery information will be collected. When the threshold V is bigger the chance of collecting the checkpoint recovery information will be less. So the failure free cost will be less. After V exceeds 0.1 the overhead is almost zero. Here, our scheme is almost equivalent to the Lazy scheme.

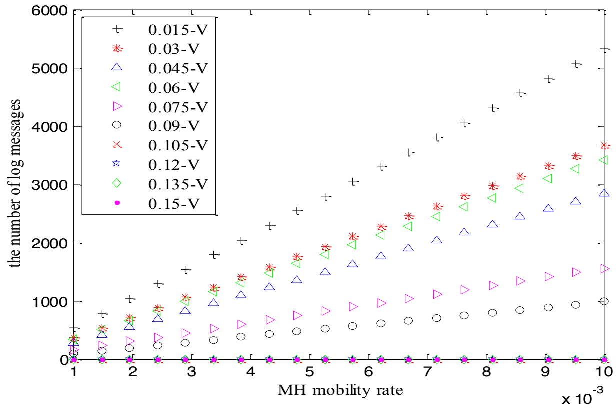

Fig.3. Failure-free Cost varies by V.

Failure-free costs imposed on each scheme are shown in Fig.4. For Lazy scheme, the number of checkpoint logging information needed to collect is 0. The proposed scheme, five simulation experiments are conducted to compare with the Frequency-based schemes by producing different random visit time (to the MSS_p) VT_i^p using the V at 0.08. For other schemes, the number of checkpoint logging information increases with the MH mobility rate. To Frequency-based scheme, when the value of k becomes zero, that is to say, after k handoffs the new MSS will perform the collection of the latest checkpoint and message logs. So the failure-free cost increases with the k value. This happens because with the increase of k the frequency of collecting the checkpoint recovery information increases, and then the failure-free cost will increase too. The Frequency-based schemes and our proposed scheme have the similar performance in some specific cases. However, the failure-free cost of our scheme is more widely dispersed than Frequency-based scheme. As illustrated above, the failure-free cost of our scheme will not definitely decrease with the threshold value V and will show flexible performance based on the actual sojourn time in the MSS.

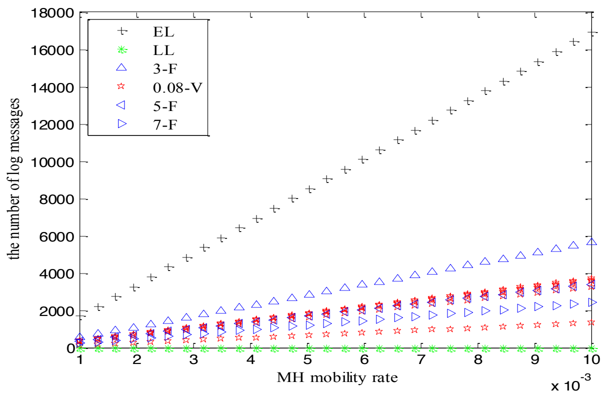

Fig.4. Failure-free Cost.

Fig.5 shows the probability of collecting recovery information from non-local MSS while the failure occurs using the equations (7)-(9). The X-axis is the simulation times, there are totally 30 simulations. In each simulation a set of visiting time is randomly generated. For Eager scheme, the probability of collecting recovery information is 0 which has the lowest failure cost. On the other side, the probability of collecting recovery information of Lazy scheme is the highest, almost approaches 1. The Frequency-based scheme generally has the second largest failure cost in all the four schemes. And compared to Frequency-based scheme our AccessPattern-based scheme always plays better performance as depicted in Fig.5. The failure-recovery cost of our scheme is also more widely dispersed than Frequency-based scheme, and equally important shows better flexibility.

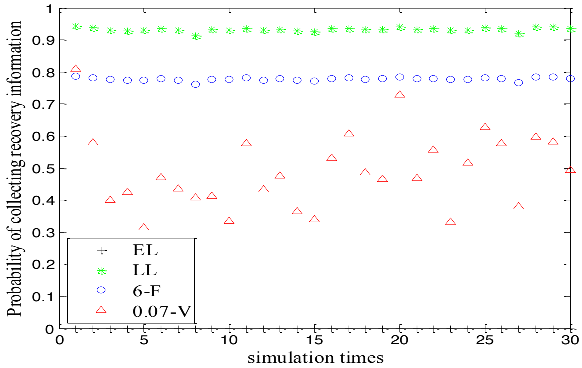

Fig.5. The Probability of Collecting Recovery Information.

6. Conclusion

The popularity of mobile applications is steadily increasing. Limited to performance of the mobile host and mobile network, failure is easy to cause in the mobile computing environment. Checkpointing can help alleviate the failure processing burden of the system and rapid recover the system. Based on the user access pattern, mainly considering the visit time of the mobile host to the sojourn mobile support stations, a checkpointing data storage scheme was proposed in this paper. Compared to Eager, Lazy and Frequency-based scheme, the number of checkpoint and message logs needed to transfer during two consecutive failures is evaluated. At the same time, the probability of collecting recovery information from non-local MSSs during failure is also calculated and simulated. The simulation results indicated that the Access-Pattern aware checkpointing data storage scheme has good performance. The scheme could be adaptable to the nomadic and mobile computing environment smoothly.

Acknowledgements

The work is fully supported by National Natural Science Foundation of China (No.61100004), and Fundamental Research Funds for the Central Universities (HEUCF100607).

References

1. Biswas S, Neogy S, Dey P. Mobility Based Checkpointing And Trust Based Recovery In Manet. International Journal of Wireless & Mobile Networks (IJWMN), 2012;4(4): 53-69.
2. Ssu KF, Yao B, Fuchs WK, Neves NF. Adaptive Checkpointing with Storage Management for Mobile Environments. IEEE TRANSACTIONS ON RELIABILITY, 1999;48(4): 315-324.
3. Perich F, Joshi A, Finin T, Yesha Y. On Data Management in Pervasive Computing Environments. IEEE Transactions on Knowledge and Data Engineering, 2004;16(5): 621-634.
4. Kiehn A, Raj P, Singh P. A Causal Checkpointing Algorithm for Mobile Computing Environments. ICDCN 2014, LNCS 8314, 2014: 134-148.
5. Wu SY, Fan HH. Activity-Based Proactive Data Management in Mobile Environments. IEEE Transactions on Mobile Computing, 2010;9(3): 390-404.
6. Pradhan DK, Krishna P, Vaidya NH. Recoverable Mobile Environment: Design and Trade-off Analysis. Proceedings of FTCS-26, 1996: 16-25.
7 Yao B, Ssu KF, Fuchs WK. Message Logging in Mobile Computing. Proceedings of FTCS 1999: 294-301.
8. Neves N, Fuchs WK. Adaptive recovery for mobile environments. Communications of the ACM, 1997;40(1): 68-74.
9. Men CG, Xu ZP, Wang DS. An Efficient Handoff Strategy for Mobile Computing Checkpoint System. 2007 IFIP International Conference on Embedded and Ubiquitous Computing, Taipei, Dec. 2007; LNCS(4808):410-421
10. Park T, Woo N, Yeom HY. An Efficient Recovery Scheme for Mobile Computing Environments. Proceedings of ICPADS 2001:53-60.
11. George SE, Chen IR, Jin Y. Movement-Based Checkpointing and Logging for Recovery in Mobile Computing Systems. MobiDE'06, June 25, 2006:51-58.
12. Park T, Woo N, Yeom H. Efficient Recovery Information Management Schemes for the Fault Tolerant Mobile Computing Systems. Proc. 2001 Int'l Conf. on Parallel and Distributed Systems, KyongJu, Korea, June 2001.
13. Park T. Region-based Recovery for Reliable Mobile Computing Environment. Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), December Edition, 2013;3(12):12-19.