Lease Based Consistency Scheme in the Web Environment

Byounghoon Lee Sunghwa Lim Jai-Hoon Kim
Graduate School of Information and Communication
Ajou University
Suwon, Republic of Korea
{componer, holyfire, jaikim}@ajou.ac.kr

Geoffrey C. Fox
Community Grids Lab
Indiana University
Bloomington, Indiana
gcf@indiana.edu

Abstract. There are many methods to maintain the consistency in the distributed computing environment. Efficient schemes for maintaining consistency should ideally take into account the following factors: lease duration of replicated data, data access pattern and system parameters. Lease method is used to supply the strong consistency in the web environment. During the proxy's lease time from web server, web server can notice the modification to the proxy by invalidation or update. In this paper, we analyze the lease protocol performance by varying update/invalidation scheme, lease duration and read rates. By using these analyses, we can choose the adaptive lease time and proper protocol (invalidation or update scheme of the modification for each proxy in the web environment). As the number of proxy for web caching increases exponentially, more efficient method for maintaining consistency should be designed. We also present 3-tier hierarchies on which each group and node independently and adaptively choose proper lease time and protocol for each proxy cache. These classifications of the scheme make proxy cache adaptive to client access pattern and system parameters.

1. Introduction

As the number of the application and users grows exponentially in the web environment, server has the message overload problem which incurs the congestion of client request and increases the response time. One approach to cope with the increment of resource utilization is to cache data near client [1]. For an example, proxy for web

caching has some advantages over reducing the overhead at the server and decreasing in response time for clients. Proxy must keep the consistency of cached data to preserve them from the stale information. Fig.1 shows web proxy architecture with lease consistency. Information broadcasting system like stock trading, weather broadcast and news is modified just at server side. Server provides the information to the proxy to reduce the latency. Consistency between server and proxy is maintained by lease and invalidation/update schemes.

There are many protocols to maintain the consistency. Lease method is used to supply the strong consistency during the lease time. In the original lease approach, the server grants a lease to each request from a proxy. The lease denotes the interval of time during which the server agrees to notify the proxy if the object is modified [2]. However lease has some drawbacks. One of principal problems of the lease is that it forces the system to keep strong consistency by notifying all modifications to a proxy. If cached data is modified frequently during the lease, message overhead would increase largely. Two conventional protocols are used to maintain data consistency for notifying the modification.

- Update scheme: When a client tries to access a data from the server, the same cached pages in every proxy are updated whenever server node modifies the data page.
- Invalidate scheme: Whenever a remote node modifies a data, the local copy is invalidated.

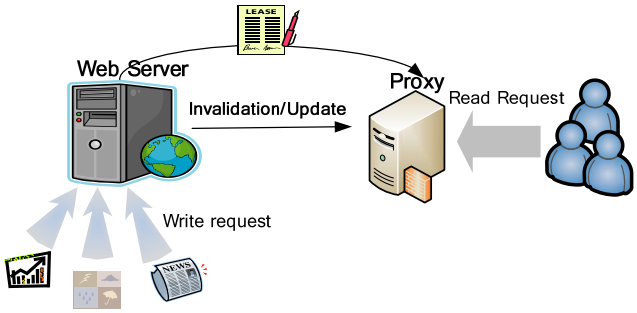

Fig. 1 Web Proxy Architecture with lease consistency

It is a critical part to choose invalidation or update for the notification method during the lease time. In this paper we analyze the notification protocol overhead of main-

taining cache consistency during the lease time. Efficient consistency maintaining schemes should ideally take into account the following factors: lease duration, data access pattern and system parameters (resources overhead such as CPU time, storage cost and communication delay). Each factor is related with each other.

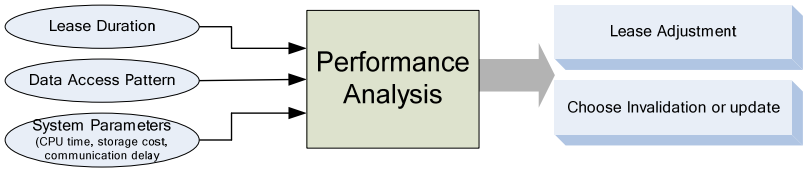

Fig. 2 Adaptive consistency maintenance flow

For an example, in general case invalidation scheme is efficient when many remote updates are occurred consecutively. But in short lease time, update scheme could be more efficient than invalidation because remaining remote updates are ignored after lease expiration time. It is interesting to analyze how lease duration, modification notice schemes (update scheme and invalidation scheme) affect on performance. Thus, in this paper we analyze performance by varying lease duration, modification scheme and local access ratio to remote modification. To reduce overhead, efficient lease duration and modification notice scheme need to be adjusted according to data access patterns and system parameters. By analyzing the cost of maintaining consistency according to lease time, shared data access pattern and system parameters, we can find proper lease time and choose alternative notice scheme of the modification. Each proxy maintains the consistency of data stored in its cache and consistency maintenance is performed efficiently by requesting the lease time and adopting proper notice scheme of the modifications from the server. We demonstrate the efficiency of our adaptive scheme through simulation as well as mathematical analysis by performance modeling. Result of mathematical analysis matches closely with the simulation results. When the number of proxy increases exponentially, more efficient consistent managing method should be designed. In this paper we also present 3 tier hierarchies, each level is divided by lease duration property and notify scheme of modification. First level is subdivided by short lease and long lease. And second level is subdivided by

invalidation and update scheme for notifying the modification. These classifications of the scheme make each proxy cache adaptive to each client access pattern.

Early papers[5,6] show the scheme that adjusts the lease duration to reduce the message overhead according to the object lifetime, frequency of the update and read request. In this paper, we argue that to find optimal notification scheme in lease duration is a critical factor to reduce the message overhead because lease duration affects to the efficient notification scheme. There is a tradeoff between data fault and local update. As lease time increases data access fault (data miss) decreases while local update cost increases by data update from server. On the other hand, as lease time decreases, cases are reversed.

Research contributions of our work are as follows :

- Our work focuses on developing a dynamic notification scheme along with read rates, read and write intensity, and to determine optimal lease duration in lease based consistency mechanism. We analyze and simulate the relation of these factors.
- Another advantage of our approach is that it applies the adaptive multi level structure classifying by notification scheme and lease duration. When the number of the proxy increases exponentially, this scheme not only reduces the message overhead but also decreases the latency time by adapting to running and system environments for each proxy.

2. Related Works

Caching by proxy is necessary to reduce the network overhead at the server. However if the object is transferred to many proxy and tries to update the object, it needs the consistency mechanism to reflect the changed data. Since web pages tend to be modified at origin servers, cached version of these pages can become inconsistent at the page change from the server. Lease, invalidation and update schemes are the common method to supply the consistency.

2.1 Invalidation and update[2,3]

If an object is modified during the lease time, server notifies the modification to the proxy. There are two methods to notify to clients. One is invalidation and the other is update. The former invalidates the proxy when it received the modification message. Therefore, subsequent read request make the proxy require the page from the server. When modification of the page occurs frequently in a short time, invalidation is very efficient. But when the object is modified infrequently, turnaround delay overhead for accessing the page increases. On the other hand, update doesn't need the turnaround delay because the object or modification part of object is sent upon each modification. This method has the drawback of the network overhead.

2.2 Competitive Update

In write-update protocol if cache data is loaded, proxy keep the cache data regardless of local data accesses. However, there is a problem that cached data block must be updated many times without local access. To resolve this problem while maintaining advantage of the write-update, local cache data should be invalidated after critical update count. Whenever a remote client requests an update, the update counter is increased. When the counter reaches to critical update count, the cache data is invalidated upon remote update[3]. Critical update counter to the each cache block could have a different value and be changed adaptively.

2.3 Adaptive Scheme

To reduce the network bandwidth and latency, invalidation and update scheme must be applied adaptively by the time-varying memory access patterns of an application [4]. Adaptive scheme chooses one of the invalidation protocol and competitive update protocol according to data access pattern and system parameters.

2.4 Lease Mechanism

Lease scheme is a time-based mechanism that maintains efficient consistent access of cached data in distributed systems [5]. The server assigns a lease on a page to every proxy and agrees to notify modifications of the page during the lease duration to the proxy. The client doesn't need to poll the server during the lease duration. It starts to poll when a read request arrives after lease expires [6].

Lease duration should be determined by considering the tradeoff between the server state space and network messages. As the lease duration become shorter, it needs lesser state space and more network messages. In contrast, it needs larger state space and needs lesser network message as the lease duration become longer.

The expiration time of the lease is applied adaptively to the proxy cache. The lease duration is determined based on various objects and system properties. The policy of the lease duration decision is classified by the object life time, client access characteristics and server state space [6].

- Age-based leases: the number of update messages can be strongly reduced compared to the case where all leases have the same expiration time
- Renewal Frequency-Based leases: the overhead can be reduced by granting longer leases to proxies that have sustained interest in the object.
- State space overhead-Based leases: As longer leases are granted, the space which is needed to maintain the state of the object becomes larger. By granting shorter leases to popular objects, the server can adaptively control the amount of state needed to maintain.

2.5 Proactive DNS cache update protocol [9]

A dynamic lease technique is used for DNS cache update protocol to keep track of the local DNS name servers that matches the clients with an Internet server. Also dynamic lease reduces communication overhead and storage overhead and makes the DNS cache update protocol lightweight. Client query rate at DNS name servers is one of factors to decide whether or not to apply leases scheme.

To make the DNS name server reliable, the DNS name server grants and maintains the leases for the DNS resource records of the Internet service. The lease duration is dependent on the domain name to IP address mapping change frequency of the specific DNS resource record.

2.6 Consistency Maintenance in Service Discovery [8]

Devices for variable services can discover their environment by using the service discovery protocols. Service discovery protocols allow devices to detect and adapt to changes of the topology. Consistency maintenance in service discovery guarantees that Users get the correct services by discovering. To maintain the consistency, the User has to subscribe either directly to the Manager (2-party subscription) or to a Registry (3-party subscription) to receive updates. A subscription between the User and the Manager or between the User and the Registry remains valid until the subscription lease does not expire. Users send messages periodically to the lessee to show the interest with the service for maintaining a valid subscription lease.

The subscription between the entities may remain valid, even though update notification fails. This is because the entities may face short-term failures, and restore connectivity before the subscription lease expires. Therefore, it needs to continue subscription process for guaranteeing Users to maintain consistency. This type of recovery is subscription-recovery. When the subscription lease expires, consistency maintenance depends on the inherent capability of the service discovery protocol to detect, and rediscover purged nodes and services. Hence, this type of recovery is called purge-rediscovery.

3. Cost Analysis for Lease-based Efficient Notification Scheme

If an object is modified during the lease duration, lease server notifies the proxy of any modification made to the object. The client is not required to poll the server during the lease duration even if read operations occurred consecutively. If a page notification which is occurred after read operations is executed by remote write operation

in lease duration, notification is processed by three methods, which are invalidation, update and competitive update. To determine which notification method should be used to maintain the consistency is a critical thing to reduce the message overhead. It is not easily manageable problem because it is based on the lease mechanism.

Early studies[3,4] showed analytical comparisons of invalidation, update and competitive update without the lease by using the segment model. We consider the cost of messages as the cost metric for the invalidation, update and competitive model. We assume that particular page P at the proxy cache is accessed by clients. These accesses can be partitioned into segments. A Segment is defined as a sequence of remote updates between two consecutive local accesses by a node. A new segment begins with the first access by a client following an update to the page by the server. Segments are defined from the point of view of each node[4]. Fig.3 shows how the segment is composed.

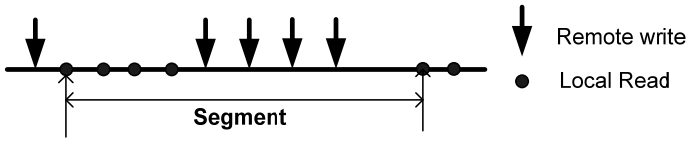

Fig.3 Definition of the Segment

Parameter for analyze the cost are shown in Table1.

Table1. Parameter for analysis

Parameters	Description
$C_{control}$	Cost to send the control message
C_{upd}	Cost to update a page of the proxy cache
C_{page}	Cost to replace a page in the proxy cache
U_{comp}	Competitive update count
u_{avg}	Average update count per segment

To compare the various notification protocol, we assume that $C_{contorl} = 5$, $C_{upd} = 10$, $C_{page} = 35$ and $U_{comp} = 4$. The cost of notification protocols in a segment are computed as follows:

Invalidate overhead cost $= (C_{control} + C_{page})$

Update overhead cost $= u_{avg} C_{upd}$

Competitive update $= u_{avg} C_{upd}$ (if Update Count $\leq$ 4)

$= u_{avg} C_{upd} + (C_{control} + C_{page})$ (if Update Count > 4)

Fig.4 shows an analytical comparison of message overhead for one segment. We assumed update count threshold 4 for competitive update protocol. When remote write occurred less than threshold count, competitive update protocol notifies to the proxy by update and when remote write occurred more than threshold count, it notifies by invalidation. Update and competitive update protocols are better choices if update count is smaller than 5. However, invalidation is the best choice when update count is greater than 5.

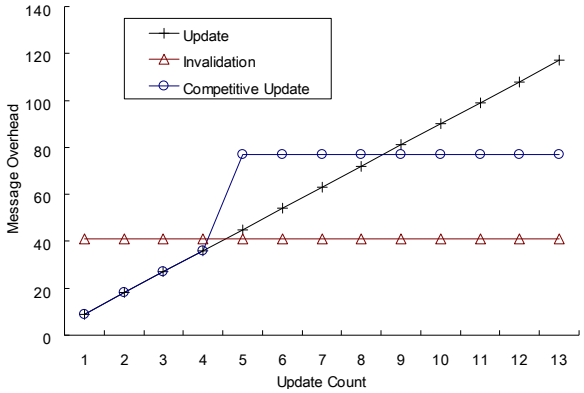

Fig.4 Compare the notification scheme

In this paper we analyze the lease cooperated with the notification scheme, which are invalidation, update and competitive update. Analysis is divided by two subjects which are general cases(section 3.1) and web burst cases(section 3.2). In the Web environment, read and remote write access to the proxy have burst property. After that, we compare the performance in section 3.3 and 3.4.

3.1 General Case Analysis of the Lease based notification scheme

We assume that read interval of each node and write interval of the remote node follow exponential distribution. In this case, it is hard to extract the special feature for deciding the lease duration or the notification scheme. To analyze the cost, we divide general case into two cases. One case is that the lease duration is longer than segment length and the other case is that the lease duration is shorter than segment length. We use Markov model for cost analysis. Fig.5 describes the meaning of the state. *X* means how many times lease renewal is accomplished and *Y* means how many times remote write is performed consecutively. Fig.6 shows the Markov model based on rate of read/write operation.

Table 2. Parameters for simulation

Parameter	Description
λ_r	Occurrence rate of local read access with exponential distribution
λ_w	Occurrence rate of remote write with exponential distribution
t_{lease}	Lease expiration time

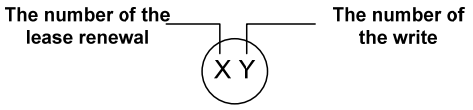

Fig.5 State description

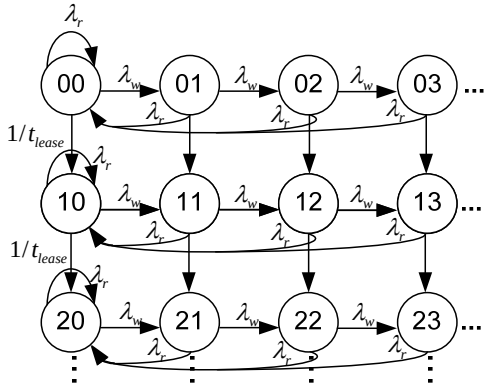

Fig. 6 Markov Chain model of the lease scheme

3.1.1 Lease duration longer than segment length

In this case we assume that more than one segment is formed during the lease duration.

Parameters to analyze the cost are shown in Table 3.

Table3. Parameters for analysis

Parameters	Description
N	The number of the segment in lease (n>1)
$C_{control}$	Cost to send the control message
C_{upd}	Cost to update a page of the proxy cache
C_{page}	Cost to replace a page in the proxy cache
U_{comp}	Competitive update count
u_{avg}	Average update count per segment
$p(k)$	Probability to be *k*'th consecutive update
σ	Standard deviation of the number of the update
$\int_{u_{comp}}^{\infty} f(u_{avg}, \sigma) du$	Probability that updates are occurred more than u_{comp}

When each segment begins with first read, proxy cache needs to obtain the page by sending a control message. If ***n*** segments are included during the lease, overhead cost for lease invalidation case is as follows :

$$\text{Lease Invalidate overhead cost} = n(C_{control} + C_{page})$$

In update case, page replacement cost is requested at the read of the first segment, and every update requirement needs the update cost.

$$\text{Lease Update overhead cost} = n(u_{avg} C_{upd}) + C_{page}$$

Cost of the lease competitive update varies by segment status in lease.

We assume that the average number of the update in a segment is u_{avg} .

If update occurs above u_{comp} for a segment, it requires C_{page} for first reading cost in the next segment.

We assume that occurrence probability of the remote write is *pdf* (probability density function) of normal distribution as follows:

$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{1}{2}(\frac{x-u_{avg}}{\sigma})^2]$ (*x*: the number of the consecutive write)

$F(x) = \int_{-\infty}^{x} f(t)dt$: *cdf*(cumulative distribution function) for the normal distribution

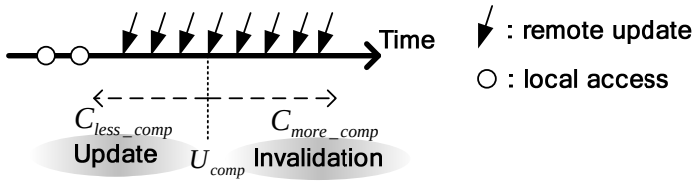

Fig.7 Competitive update cost U_{comp} =4

Fig.7 shows how total cost of the competitive update is composed. Overhead cost for lease competitive as follows:

Lease Competitive Overhead cost = $n(C_{more\_comp} + C_{less\_comp})$

C_{more_comp} is the cost of case that the number of the remote write is larger than U_{comp} and C_{less_comp} is the cost of case that the number of the remote write is less than U_{comp} . The probabilities that an update occurs above U_{comp} is $\int_{u_{comp}}^{\infty} f(x)dx$

Therefore we can compute C_{more_comp} and C_{less_comp} as follows :

$$C_{more\_comp} = \int_{u_{comp}}^{\infty} f(x)dxC_{page}$$

$$C_{less\_comp} = \sum_{x=1}^{x=u_{comp}} xF(x)C_{upd}$$

3.1.2 Lease duration shorter than segment length

We define system parameters to make a cost expression as shown in Table 4.

Table 4. Analysis Parameters for lease based invalidation

Parameter	Description
λ_r	Occurrence rate of local read access with exponential distribution
λ_w	Occurrence rate of remote write with exponential distribution
$1-e^{-(\lambda_r+\lambda_w)t_{lease}}$	Probability that either local read or remote write occur before t_{lease}
$\frac{\lambda_w}{\lambda_r+\lambda_w}(1-e^{-(\lambda_r+\lambda_w)t_{lease}})$	Probability that remote write is occurred before t_{lease}
$T(\lambda_{r,w})$	Elapsed time from the lease start to the λ_r or λ_w occurrence
$INV_{cost}(t)$	Invalidation cost during t_{lease} - t

3.1.2.1 Invalidation

In this case, overhead cost of the invalidation is composed with the condition of occurring the write before lease expiration and after lease expiration.

- Cost when remote write is occurred before t_{lease} (Fig. 8)

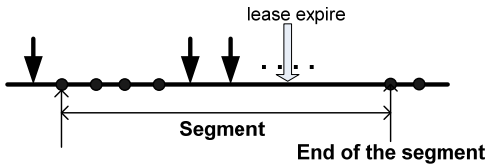

Fig.8 Lease expiration in remote write

Irrespective of the number of the update, there is the only page replacement overhead, ($C_{control} + C_{page}$). Probability that an update occurs more than once is

$$\frac{\lambda_w}{\lambda_r + \lambda_w}(1 - e^{-(\lambda_r + \lambda_w)t_{lease}})$$

Thus the cost of the lease based invalidation is calculated as follows

$$INV_{cost}(0) = \frac{\lambda_w}{\lambda_r + \lambda_w}(1 - e^{-(\lambda_r + \lambda_w)t_{lease}})(C_{control} + C_{page})$$

- Cost when remote write is not occurred before t_{lease}

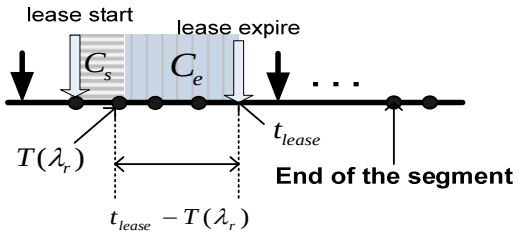

Fig. 9 Lease expiration before remote write

Cost of the invalidation is divided into two parts. First part is the cost(C_s) between lease start and local access event. Second part is the cost(C_e) between local access event and lease expiration. Second part is recursively composed with successive local access. Fig. 9 shows both cases of the cost.

$$INV_{cost}(0) = C_s + C_e$$

Probability that local read occur before t_{lease}

$$(1 - e^{-(\lambda_r + \lambda_w)t_{lease}})(\frac{\lambda_r}{\lambda_r + \lambda_w})$$

So C_s is calculated as

$$C_s = (1 - e^{-(\lambda_r + \lambda_w)t_{lease}})(\frac{\lambda_r}{\lambda_r + \lambda_w} C_{control})$$

C_e is the cost that is iterated between $T(\lambda_r)$ and t_{lease}.

$$C_e = INV_{cost}(t_{lease} - T(\lambda_r))$$

3.1.2.2 Update

Probability that remote write is occurred *x* times consecutively before t_{lease} is as follows:

$$\int_0^{t_{lease}} \lambda_w e^{-\lambda_w t_{lease}} dt = 1 - e^{-\lambda_w t_{lease}}$$

Probability that remote write occurs in lease duration is

$$p_w = 1 - e^{-\lambda_w t_{lease}}$$

Probability that remote write doesn't occur in lease duration is

$$\overline{p_w} = 1 - p_w = e^{-\lambda_w t_{lease}}$$

Probability that first remote write occurs in first lease duration is

$$p_w(0) = 1 - e^{-\lambda_w t_{lease}}$$

Probability that first remote write occurs in second lease duration is

$$p_w(1) = \overline{p_w} p_w = e^{-\lambda_w t_{lease}} (1 - e^{-\lambda_w t_{lease}})$$

We can yield the equation as follow

$$p_w(n) = (\overline{p_w})^n p_w = e^{-n\lambda_w t_{lease}} (1 - e^{-\lambda_w t_{lease}})$$

$p_w(n)$ means probability that first remote write is occurred after *n* count lease duration.

Update overhead cost is as follows:

$$UP_{cost} = \sum_{x=0}^{\infty} x p_w(x) C_{control} + U_{avg} C_{upd}$$

3.2 Burst Case Analysis of the Lease based notification scheme at Web environment

The Web server has the property that read or write are occurred successively if the read/write operation is once started. For an example, news site updates the contents at specific time and a subscriber reads the news not desultorily but intensively.

3.2.1 Lease based update

Message cost varies by the lease expiration condition. Update after consecutive local reads incurs C_{page}, which is the page replacement cost. Then it produces the update cost C_{upd} on each update. Updates occurred after lease expiration are excluded from the overhead cost. Besides, entire update cost is excluded if lease is expired before first update occurs. To compute the lease based update cost, UP_{lease}, it needs to add the lease renewal costs that are generated on each lease request.

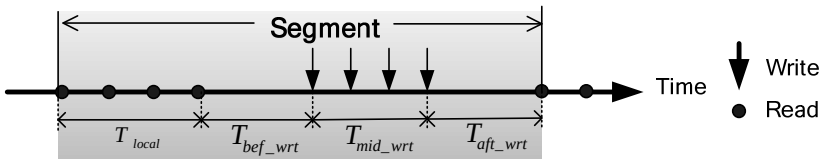

Fig. 10 Lease expiration case at the segment

Fig. 9 shows how the segment area is separated. Lease is expired at any moment(T_{local}, T_{bef_wrt}, T_{mid_wrt} and T_{aft_wrt}) in a segment. Therefore, lease based update cost per time is composed with four cost event cases. Each case is classified with lease expiration timing. First case is expired at T_{local} that is occurred in the middle of consecutive local access. Second case is expired at T_{bef_wrt} that is occurred before the remote write. Third case is expired at T_{mid_wrt} that is occurred in the middle of consecutive remote write. Last case is expired at T_{aft_wrt} that is occurred between last remote write and first local read of next segment. Total time of the segment is composed as follows

$$T_{seg} = T_{local} + T_{bef\_wrt} + T_{mid\_wrt} + T_{aft\_wrt}$$

Table 5. Analysis Parameter for Burst case

Parameter	Description
P_{local}	T_{local} period ratio in a segment
P_{bef_wrt}	T_{bef_wrt} period ratio in a segment
P_{mid_wrt}	T_{mid_wrt} period ratio in a segment
P_{aft_wrt}	T_{aft_wrt} period ratio in a segment
W	Average remote write count happened in a segment
C_{local}	Cost occurred during T_{local} period
C_{bef_wrt}	Cost occurred during T_{bef_wrt} period
C_{mid_wrt}	Cost occurred during P_{mid_wrt} period
C_{aft_wrt}	Cost occurred during P_{aft_wrt} period
$\hat{w}$	The number of consecutive write group (same meaning with the segment)

Overhead weight of each event case varies by event period ratio and each period ratio is calculated as follows :

$$P_{local} = \frac{T_{local}}{T_{seg}}$$

$$P_{bef\_wrt} = \frac{T_{bef\_wrt}}{T_{seg}}$$

$$P_{mid\_wrt} = \frac{T_{mid\_wrt}}{T_{seg}}$$

$$P_{aft\_wrt} = \frac{T_{aft\_wrt}}{T_{seg}}$$

Total cost of lease based update is derived from joining all cases together and is calculated as follows:

$$UP_{lease/time} = C_{local}P_{local} + C_{bef\_exp}P_{bef\_wrt} + C_{mid\_exp}P_{mid\_wrt} + C_{aft\_wrt}P_{aft\_wrt}$$

At the C_{bef_wrt}, it doesn't need the update cost because lease is expired before the update, so average update cost is subtracted from the page replacement cost with the P_{rest} / 2 ratio. Lease based update cost is calculated as follows :

$$C_{bef\_wrt} = \frac{C_{page} - u_{avg} C_{upd}}{2 \max(t_{lease}, t_{seg})}$$

C_{local} is the overhead cost to request more lease time when local access still remains after lease expiration.

$$C_{local} = \frac{C_{control}}{t_{lease}}$$

C_{mid_exp} removes the ignored update overhead from the page replacement cost and it is calculated by subtracting C_{ign} from C_{page}. Details of the C_{mid_exp} is illustrated in Fig. 11.

$$C_{mid\_exp} = \frac{C_{page} - C_{ign}}{\max(t_{lease}, t_{seg})}$$

C_{ign} is the ignored cost which is occurred after lease expiration.

$$C_{ign} = \max(\frac{u_{avg}}{2}, \frac{t_{seg} P_{remote} - t_{lease} / 2}{t_{seg} P_{local}} u_{avg}) C_{upd}$$

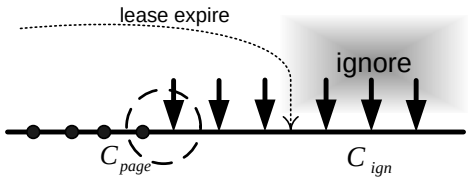

Fig.11 C_{mid_exp} details

$\frac{t_{seg}P_{remote} - t_{lease}/2}{t_{seg}P_{local}}$ means the ratio that an update would occur after lease expired.

T_{aft_wrt} area includes all remote writes in a segment. Average remote write count per segment is w, so C_{aft_wrt} is WC_{upd}

Then we can compute the cost per segment as follows :

$$UP_{lease/seg} = \frac{UP_{lease}/t}{\hat{w}}$$

3.2.2 Lease based invalidation

Parameters to analyze the lease based invalidation cost INV_{lease} per time is described in Table 6.

Table 6. Analysis parameters for invalidation case

Parameter	Description
P_{local}	T_{local} period ratio in a segment
P_{bef_inv}	T_{bef_inv} period ratio in a segment
P_{mid_inv}	T_{mid_inv} period ratio in a segment
C_{aft_inv}	Cost when the lease is expired after the consecutive remote write
C_{bef_inv}	Cost when the lease is expired before the first remote write
C_{local}	Lease renewal cost per time
$\hat{w}$	The number of consecutive write group (same meaning with the number of segment)

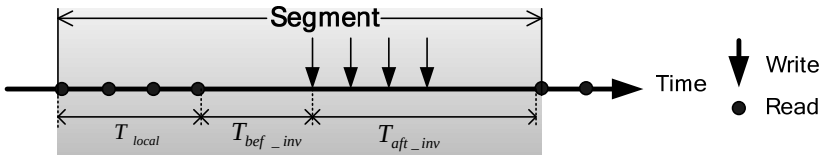

Fig.12 Lease expiration case at the segment

Segment is composed as shown in Fig.12.

$$T_{seg} = T_{local} + T_{bef\_inv} + T_{aft\_inv}$$

Overhead weight of each event case varies by event period ratio. Each period ratio is calculated as follows :

$$P_{local} = \frac{T_{local}}{T_{seg}}$$

$$P_{bef\_inv} = \frac{T_{bef\_inv}}{T_{seg}}$$

$$P_{aft\_inv} = \frac{T_{aft\_wrt}}{T_{seg}}$$

Total cost of lease based update is derived from joining all cases together and it is calculated as follows :

$$INV_{lease} = C_{local}P_{local} - C_{bef\_inv}P_{bef\_inv} + C_{aft\_inv}P_{aft\_inv}$$

C_{local} is the overhead cost to request more lease time when local accesses are still remained at proxy after lease expiration.

$$C_{local} = \frac{C_{control}}{t_{lease}}$$

If lease is expired before the remote write, it is not needed to check the consistency. Thus control cost per time is removed from the total cost.

$$C_{bef\_inv} = \frac{C_{control}}{2\max(t_{lease}, t_{seg})}$$

If invalidation for remote write is occurred more than once, server should replace the page after sending the control message.

$$C_{aft\_inv} = \frac{\hat{w}(C_{control} + C_{page})}{\max(t_{lease}, t_{seg})}$$

Invalidation cost per segment is as follows :

$$INV_{lease/seg} = \frac{INV_{lease}}{\hat{w}}$$

3.3 Performance Comparison

3.3.1 General Case Simulation for the Lease based notification scheme

Lease invalidation is efficient when the remote write ratio is larger than local read ratio, because the updates of the proxy cache don't need to be transferred until read request is received. Low frequency of the local read accesses decreases the frequency of the lease renewal and incurs small cost of the read request. When frequent remote write and infrequent local reads are happened, invalidation cost is fixed and lease renewal cost is not a burden to the proxy cache. Lease based invalidation scheme is the most efficient scheme when read rates is low. Update scheme is inefficient in low read rates because update scheme increases the overhead in proportion to the update count. Simulation conditions are described in table 7.

Table 7. Default Parameters for simulation

Parameter	Value
Lease time	200
Local access interval	40
$C_{contorl}$	5
C_{update}	10
C_{page}	40
U_{comp}	4
Segmentation Count	9000

Fig. 13 shows that lease based invalidation scheme is the most efficient at low read rates. Low read rates means that remote write is occurred frequently and local read is occurred infrequently. In this case, invalidation scheme is efficient because update scheme requires overhead in proportion to update counts.

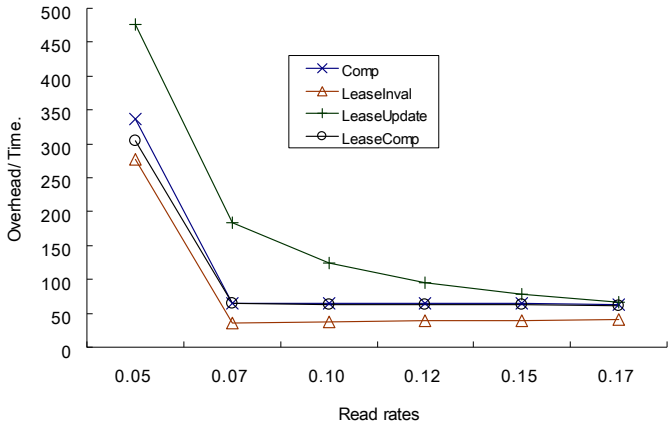

Fig. 13 Overhead of low read rates

Fig. 14 shows that lease based update scheme is the most efficient at high read rates. High read rates means that remote write is occurred infrequently while local read is occurred frequently. In this case, update is efficient because the proxy doesn't need to request the page replacement for infrequent remote write.

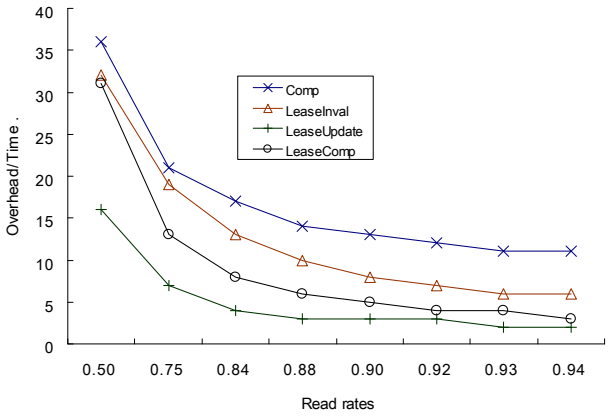

Fig. 14 Overhead of high read rates

Fig. 15 shows all the cases for lease duration and notification schemes with read rates. As read rates increases, frequency of the object modification is decreased. We can see two consistency properties from Fig. 15. One of consistency properties is that update scheme shows better performance by reducing the message overhead as remote writes happened infrequently. The other property is related with the lease scheme. As read request occurs frequently, lease scheme shows better performance than other consistency maintenance schemes without lease.

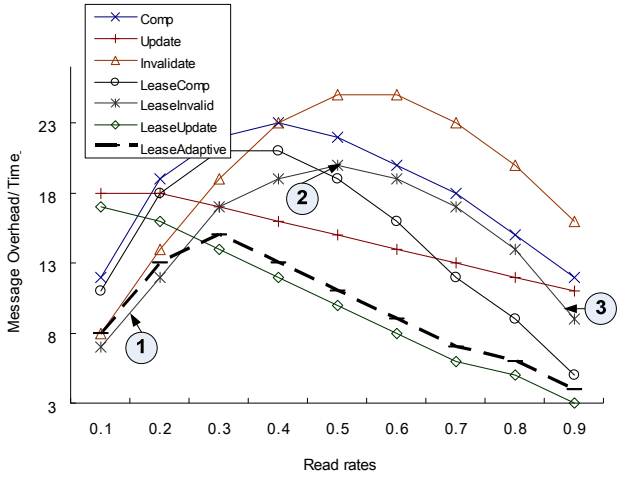

Fig. 15 Overhead of read rates

Overhead of the Lease based invalidation increases in accordance with the number of segments. The number of segments has the maximum value at 50% of the read rates. Thus the overhead increases till 50% of read rates, after that point overhead decreases. Page replacement patterns at the lease invalidation protocol are varied by the read rates and affect to the cost. Fig. 16 shows the page replacement patterns.

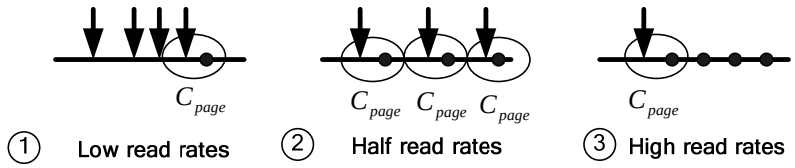

Fig. 16 Page replacement occurrence case with read rates.

(C_{page} : cost for page replacement)

Fig. 15 shows that lease based update is the best when read rates is smaller than 0.3 and lease based invalidation is best read rates is larger than 0.2. If we can predict the access pattern by analyzing, we can apply the notification scheme adaptively to the proxy cache. Analyzing for access pattern incurs the overhead to process the data. But overhead for analyzing is trivial compared with the overhead which is reduced by adaptive notification scheme.

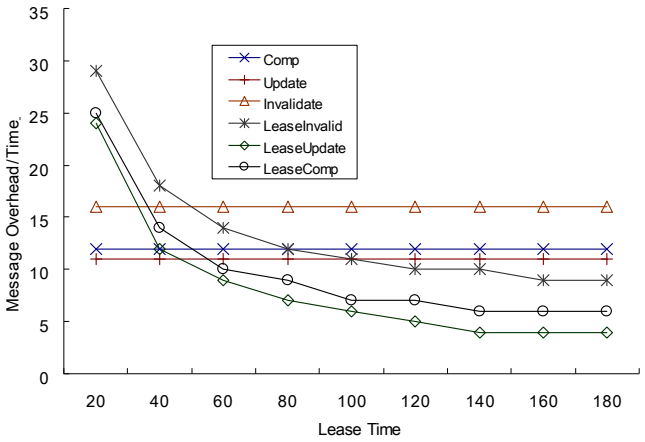

Fig. 17 Overhead of Lease duration (read rates : 90)

Fig.17 shows that lease based update scheme has the least message overhead among various consistency schemes when lease time is above 50. When read requests are occured more frequently than write request, lease based scheme is very efficient.

3.3.2 Burst Case Simulation of the Lease based notification scheme in Web environments

Lease duration should be determined by considering the tradeoff between the lease renewal overhead and update overhead. As the lease duration become small, the lease renewal overhead become large and the update overhead become small. In contrast, as the lease duration becomes large, the lease renewal overhead becomes small and the update overhead become large.

Lease based update simulation is composed with some factors. Table 8 describes the simulation parameters.

Table 8. Simulation Parameters

Parameter	Description
P_{local}	Ratio that lease is expired in the middle of consecutive local access
P_{bef_wrt}	Ratio that lease is expired before the remote write
P_{mid_wrt}	Ratio that lease is expired in the middle of consecutive remote write
P_{aft_wrt}	Ratio that lease is expired between last remote write and first local read of next segment

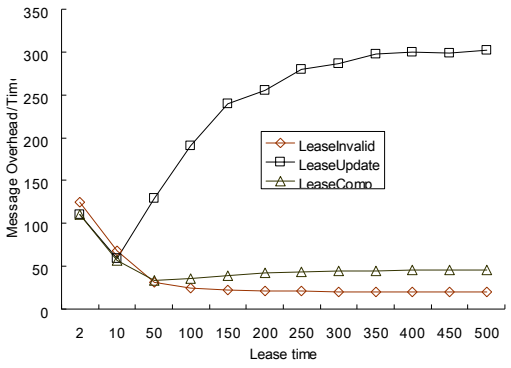

Fig. 18 Message Overhead of the three lease based scheme

(Ratio of periods are P_{local} :30%, P_{bef_wrt} :20%, P_{mid_wrt} :30%, P_{aft_wrt} :20%)

Fig. 18 shows that message overhead decrease in lease based invalidation scheme until lease time is less than 10. Because larger the lease duration, it makes lesser the lease renewal overhead with increasing little update. But when lease time is above 10, it incurs a number of the update with reducing little renewal overhead. At the lease base invalidation case, proxy cache needs to replace page just once regardless of the number of remote update during lease time. So Fig. 18 shows that message overhead is fixed around 20 after lease time 100. Lease based competitive is similar with lease based invalidation but it has the gap of the overhead for competitive update count .

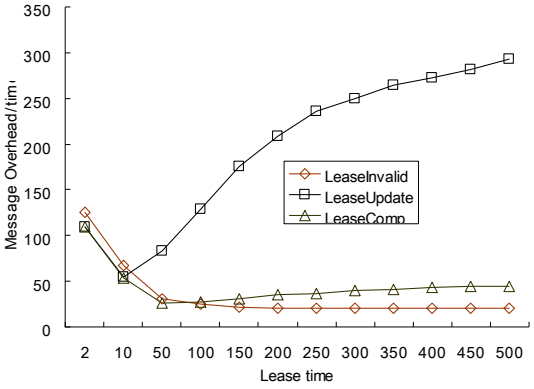

Fig. 19 Message Overhead of the three lease based scheme

(Ratio of periods are P_{local} :21%, P_{bef_wrt} :29%, P_{mid_wrt} :22%, P_{aft_wrt} :28%)

Fig.19 shows that the lease update cost increase more slowly than Fig.18. As P_{bef_wrt} become larger and P_{mid_wrt} become small, it decreases the remote write count that is included during lease duration. Reducing update count during the lease time decreases the message overhead. Lease invalidation and lease competitive is similar with the Fig. 18.

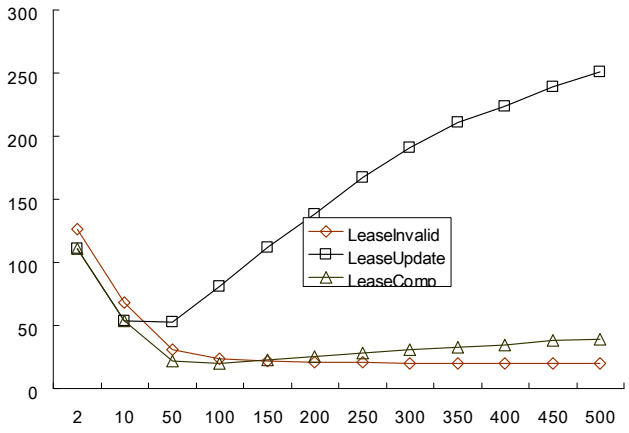

Fig. 20 Message Overhead of the three lease based scheme

(Ratio of periods are P_{local} :14%, P_{bef_wrt} :36%, P_{mid_wrt} :14%, P_{aft_wrt} :36%)

Fig. 20 shows that message overhead of the lease update increases more slowly than Fig. 19. Lease invalidation and lease competitive scheme shows similar performances with the Fig. 19.

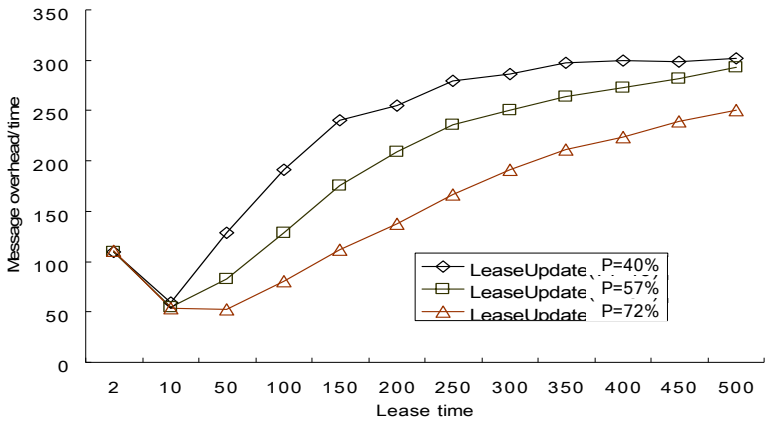

Fig. 21 Message Overhead for the variable period ratio ($P = P_{bef\_wrt}$)

Fig. 21 shows that update overhead cost increases rapidly as P_{bef_wrt} decreases.

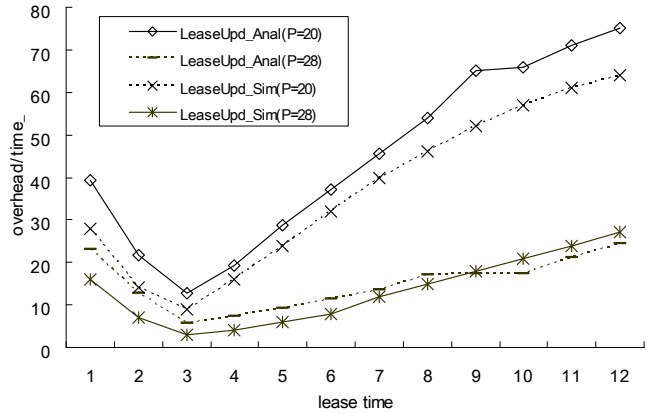

Fig. 22 Compare analysis with simulation ($P = P_{bef\_wrt}$)

Fig.22 shows that our analysis for lease based update approximates our simulation results. It means that lease based notification scheme has the proper lease time and

employ proper modification notifying scheme dynamically for efficient consistency. To achieve the efficient consistency, it needs to determine the lease time and choose the notification scheme adaptively. To improve the efficiency, it needs to classify the server group by lease duration and notification scheme and to make the hierarchical structure to apply the proxy adaptively by server group.

3.4 Adaptive scheme

Above simulation results show that lease duration and notification scheme need to be adaptive according to the various patterns of server access and request patterns of client. If cache data has the property of periodic modification and read access pattern, we can apply the consistency maintaining scheme efficiently by analyzing the sample period of the cache data. Lease duration and notification scheme is determined adaptively by predicted data access pattern.

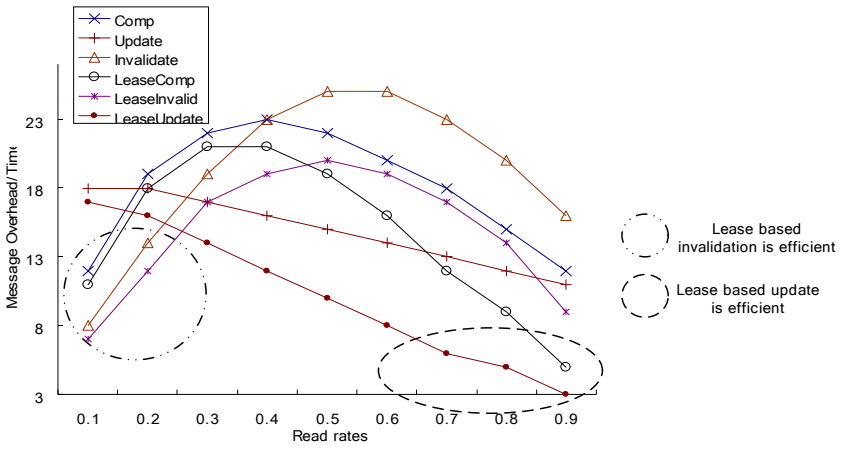

Fig. 23 Efficient notification scheme along with the read rates

Fig.23 shows that invalidation is more efficient at low read rates. On the other hand, update is more efficient at high read rates. So if we make the adaptive notification group by read rates, we can maintain the consistency with less overhead. Through the analysis, we can choose suitable lease time and notification method.

4. Hierarchical Consistency maintenance

When the number of proxy increases exponentially, more efficient consistent managing method should be designed. In this paper we present 3 tier hierarchies in which each level is divided by lease duration property and notification scheme. First level is subdivided by short lease and long lease. Second level is subdivided by invalidation and update for notifying the modification. These classifications of the scheme make proxy cache adaptive to client access pattern

4.1 Notification Method Based Consistency Hierarchy

A server handles a lot of proxy caches in the web. Therefore, it is necessary to reduce network bandwidth for transferring the cache object modification. 2-tier scheme is efficient for many proxy caches. Members of the group is classified by the efficient notification scheme. Proxy leader broadcasts the modification to the group member. Efficient notification method to the proxy cache is determined by the remote access pattern. When a number of the consecutive remote write accesses without local read are happened, invalidation will be efficient. On the other hand updates scheme will produce better performance for frequently requested objects with infrequent remote write.

Table 9. Parameter for adaptive group

Parameters	Description
U_{upd}	Average update count for update proxy
U_{inv}	Average update count for invalidation proxy
$\boldsymbol{P}$	The number of the entire proxy
R	Update/Invalidation proxy group ratio ($\frac{N_{upd}}{N_{upd}+N_{inv}}$)
N_{upd}	The number of update proxy (PR)
N_{inv}	The number of invalidation proxy (P(1-R))
T_{update}	Total cost when group sever broadcast by update(2-tier update)
$T_{invalid}$	Total cost group when sever broadcast by invalidation(2-tier invalidation)

- **Case that all proxies are included in single group**

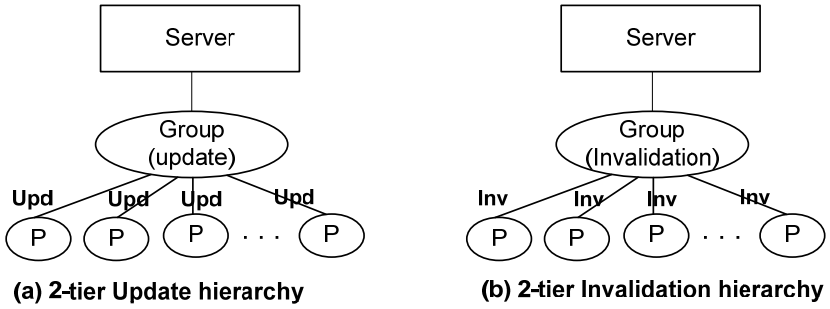

Fig.24 2-tier Notification structure by homogeneous group

2-tier update structure applies the update scheme to all the proxy including invalidation proxy. Update proxy and invalidation proxy has different average write count as U_{upd} and U_{inv}.

$$T_{update} = U_{upd} N_{upd} C_{update} + U_{inv} N_{inv} C_{update}$$

N_{upd} and N_{inv} is substituted for $P \cdot R$ and $P \cdot (1-R)$

$$T_{update} = U_{upd} \cdot P \cdot R \cdot C_{update} + U_{inv} \cdot P \cdot (1-R) \cdot C_{update}$$

$T_{invalid}$ is calculated with regardless of the remote write count. It just needs the page replacement cost.

$$T_{invalid} = (C_{control} + C_{page}) \cdot P$$

- **Notification adaptive group division case**

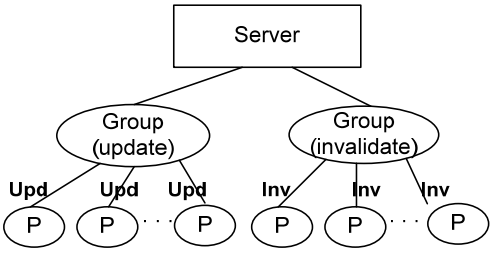

Fig.25 2-tier Notification structure by heterogeneous group

Total cost of the adaptive group sums up the update group cost and invalidation group cost.

$$T_{adap} = U_{upd} \cdot N_{upd} \cdot C_{update} + N_{inv} \cdot (C_{control} + C_{page})$$

$$T_{adap} = U_{upd} \cdot P \cdot R \cdot C_{update} + P \cdot (1 - R) \cdot (C_{control} + C_{page})$$

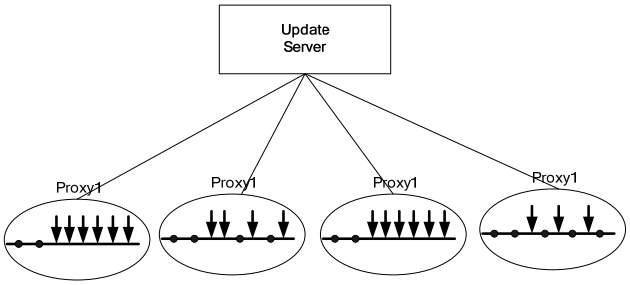

Fig. 26 Update proxy rates 50%

Fig.26 shows an example where proxy for update and proxy for invalidation are mixed. In this case, employing only one notification method(update in this example) statically from the server would be inefficient even if broadcasting form the server to proxies can decrease the message overhead. So it needs to broadcast by using adaptive notification server like Fig. 27.

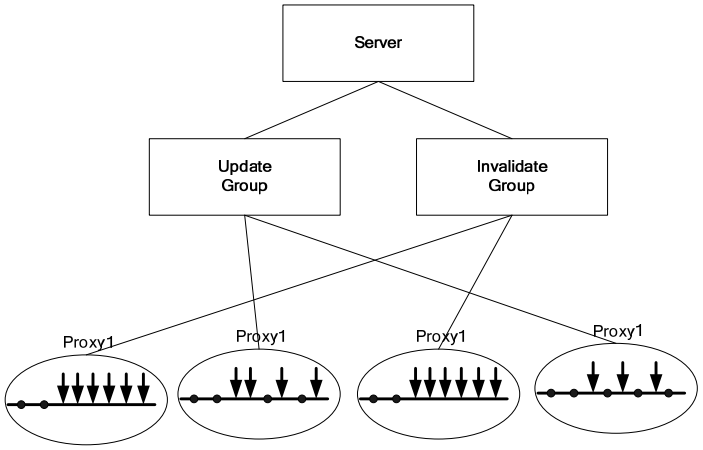

Fig. 27 Notification by adaptive group

We define system parameters to analyze as shown in Table 10.

Table 10. Simulation parameter for adaptive group

Parameters	Value
U_{upd}	2
U_{inv}	6
P	100
C_{update}	9
C_{page}	40
$C_{contorl}$	1

Fig. 28 shows that adaptive group server is more efficient than single notification server employing only one of invalidation or update.

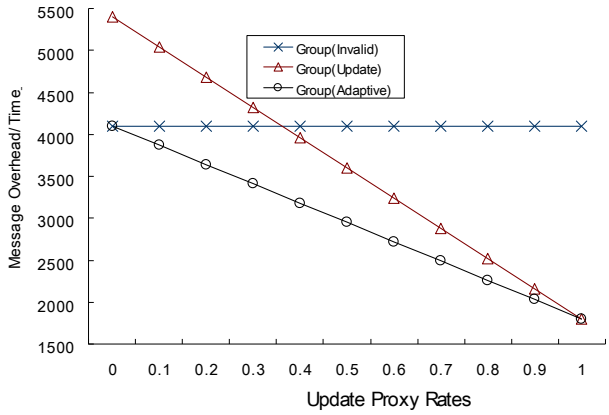

Fig. 28 Compare the notification structures
(by increasing update proxy group rates)

4.2 Lease Duration Based Consistency Hierarchy

We can determine the duration of the lease considering a tradeoff between the server state space and network messages. As the lease duration becomes short, the state space becomes small while the number of network messages becomes large. In con-

trast, as the duration of the lease becomes large, the state space at the server becomes large and the number of network messages becomes small.

Group of the Proxy cache is determined by the property of the proxy cache.

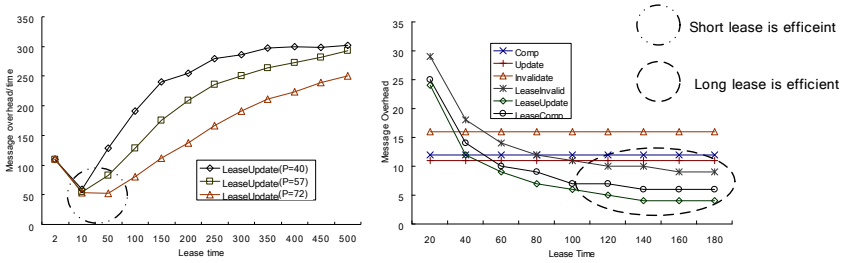

Fig. 29 Lease adjustment by data access pattern($P_r = P_{bef\_wrt}$)

From Fig.30, we see that efficient lease time varies by data access pattern.

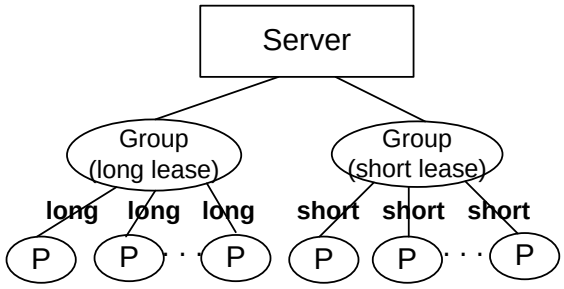

Fig.30 2-tier Lease duration structure

4.3 Complex Consistency Hierarchy

Modification pattern property of the cache object affects on the notification scheme and local read access pattern affects on the lease duration. In the web environment, there are various servers and great number of proxies for the client. Each proxy cache would have specific consistency property like infrequent remote write and frequent local read. The purpose of this paper is also to offer adaptive structure of a proxy by complex consistency scheme.

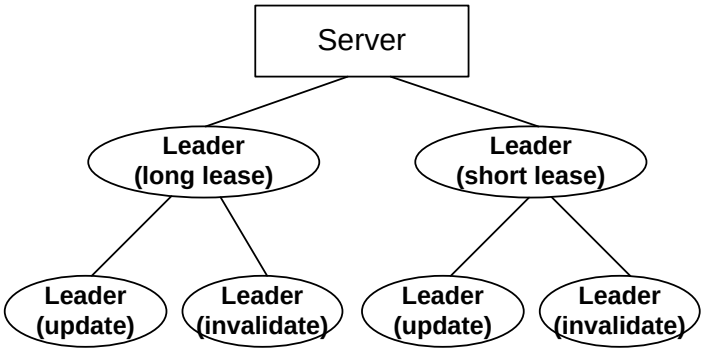

Fig.31 3-tier Structure for Complex consistency

Each proxy cache joins in the most suitable group with considering the local read pattern and remote write pattern. Complex consistency structure with 3-tier form offers the sufficient group for consistency property**.**

5. Conclusion

In this paper we analyze the notify protocol overhead to maintain cache consistency during the lease time. While previous studies have addressed in lease duration and just compared the update and invalidation, we analyze the performance combined with lease duration, modification scheme and local access ratio to remote modification. By analyzing the access pattern, we can employ the adaptive lease time and alternative notice scheme of the modification.

We also present 3-tier hierarchies in which each level is divided by lease duration property and notify scheme of modification. First level is subdivided by short lease and long lease. Second level is subdivided by invalidation and update for notifying the modification. These classifications of the scheme make proxy cache adaptive to client access pattern.

6. References

[1] J. Gwertzman and M. Seltzer, “World-wide web cache consistency” in Proc. 1996 USENIX Tech. Conf. San Diego, CA, Jan. 1996.

[2] A. Ninan, P. Kulkarni, P. Shenoy, K. Ramamritham, and R. Tewari. “Scalable Consistency Maintenance in Content Distribution Networks Using Cooperative Leases”. IEEETKDE, July 2003.

[3] Hakan Grahn, Per Stenstrom and Michel Dubois. “Implementation and evaluation of update-based cache protocols under relaxed memory consistency models”. Future Generation Computer Systems, 11(3), June 1995

[4] J.-H. Kim and N. H. Vaidya, “A cost-comparison approach for adaptive distributed shared memory” in *Proc. of 1996 International Conference on Supercomputing*, pp. 44-51, May. 1996.

[5] C. Gray and D. Cheriton. “Leases: An efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency” in *Proc. of Twelfth ACM Symposium on Operating Systems Principles*, pp. 202-210,1989.

[6] V. Duvvuri, “Adaptive Leases: A Strong Consistency Mechanism for the World Wide Web” MS thesis, Univ. of Mass., Jun. 1999.

[7] K. S. Byun, S. H. Lim and J.-H. Kim, “Two-Tier Checkpointing Algorithm Using MSS in Wireless Network” *IEICE transactions on Communications* Vol. E86-B No. 7 pp. 1-7, Jul. 2003.

[8] Sundramoorthy, V., J. Scholten, P.G. Jansen and P.H. Hartel, “On Consistency Maintenance In Service Discovery”, *4th Int. Conf. on Information, Communications and Signal Processing and 4th IEEE Pacific-Rim Conf. On Multimedia*, Vol. 3, IEEE Computer Society Press, Los Alamitos, California, 2006.

[9] Xin Chen, Haining Wang, Shansi Ren and Xiaodong Zhang, “Maintaining Strong Cache Consistency for the Domain Name System”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 8, Aug. 2007.