Log′ version vector: Logging version vectors concisely in dynamic replication

Hyun-Gul Roh\textsuperscript{a,∗}, Myeongjae Jeon\textsuperscript{b}, Euiseong Seo\textsuperscript{c}, Jinsoo Kim\textsuperscript{d}, Joonwon Lee\textsuperscript{d}

\textsuperscript{a} Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea
\textsuperscript{b} Department of Computer Science, Rice University, Houston, TX, United States
\textsuperscript{c} School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology, Ulsan, Republic of Korea
\textsuperscript{d} School of Information and Communication Engineering, Sungkyunkwan University, Suwon, Republic of Korea

A R T I C L E   I N F O

Article history:
Received 1 December 2009
Received in revised form 29 April 2010
Accepted 30 April 2010
Available online 6 May 2010
Communicated by M. Yamashita

Keywords:
Distributed computing
Distributed systems
Version vector
Log′ version vector
Replication system

A B S T R A C T

In a replication system, version vectors are logged with replicas to detect conflicts among operations. Dynamic replications where replicas are frequently created and destroyed suffer from expensive logging overhead caused by inactive entries of version vectors. Although the rigmarole of pruning vectors can delete inactive entries, the vectors may be incompatible without additional information, which also causes another overhead. This paper proposes a novel version vector called Log′ (log-prime) consisting of only three entries. By encoding based on the characteristics of prime numbers, Log′ version vectors of fixed size can be logged concisely with no pruning technique at a little sacrifice in accuracy. Simulation studies show that Log′ version vectors are accurate enough to detect almost all conflicts in the replication systems where all replicas are fully synchronizing.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

In diverse replication systems \cite{1–5}, replicas operate version vectors to enumerate distributed operations and log a large number of vector values, each of which captures individual operation, in order to track updates of replicas or to detect/resolve conflicts arising from distributed execution. Therefore, many replication systems have been suffering from the logging overhead by version vectors \cite{3–5}. The transmission overhead can be reduced by some methods \cite{6,7} primarily devised for the vector clock, which is inherently similar to the version vector in the operating rules and detecting mechanisms \cite{8,9}. However, these methods have no effect on reducing the logging overhead of version vectors that is intrinsically related to the vector sizes. Even though the approaches in \cite{3} and \cite{4} claim to reduce the logging overhead, they are ineffective when vectors are to contain many inactive entries in dynamic replications in which replicas are frequently created and destroyed.

Indeed, for the compatibility of vectors, the entries representing destroyed (retired) replicas should be preserved, though they will remain inactive. To reduce the logging overhead however, some research efforts have tried to remove inactive entries through the vector pruning, but reckless pruning results in serious side-effects, as explained in \cite{10}. Fig. 1 shows a simple case where an inactive entry is deleted immediately. If vectors are anonymous arrays, they might be corrupted due to the incompatibility of the vectors. To avoid such incompatibility, several systems \cite{11–13} enforce an agreement on the vector pruning usually through distributed consensus protocols, which involve great cost and complexity. Although replicas may manage to update vectors without deleted entries by tagging each vector entry, any two vectors representing different sets of
replicas are still incompatible because the pruning makes vectors lose necessary information after all.

To ensure compatibility of pruned vectors, Richard [14] and Wang et al. [15] individually introduced vector pruning techniques that prepare lists of final vector values of the destroyed replicas. These techniques are lightweight because they only require a retiring replica to notify others of its own retirement instead of costly consensus protocols. However, if a replica crashes without explicit notification, it is impossible to prune vectors. Saito [10] suggested a unilateral pruning technique that makes use of a physical clock for each entry. Although this can delete inactive entries without notification after fairly long expiration periods (e.g., a month), maintaining physical clocks incurs considerable overhead.

In fact, all pruning techniques demand additional information, such as entry tags or vector descriptions, which are as follows: (1) $L_0$, (2) $L_j$ is an integer value shared by all replicas, and (3) $\vec{v}_i$ is a vector that has one entry for $\forall p_k \in P$ such that $P$ is a set of prime numbers of all the replicas that have/had participated in the replication. For the three operation types generally considered in replication systems, i.e., internal, sync, and rsync operations [1], the operating rules on $PV_i = [p_i, L_0, \vec{v}_i]$ at $R_i$ are as follows:

- Let $isSent$ be an internal boolean variable.

**Rule 0** Initial value:
1. $p_i :=$ a prime number unique to $R_i$,
2. $L_0 := 0$, $\forall p_k \in P: \vec{v}_i[p_k] := 0$, $isSent := false$;

**Rule 1** Before executing an internal operation:
1. Execute $DA$;
2. $L_0 := L_0 + 1$;

**Rule 2** Before sending a sync operation:
1. Execute $DA$;
2. $L_j := L_j + 1$, $isSent := true$;

**Rule 3** Before executing an rsync operation received from $R_j$ with $PV_j = [p_j, L_j, \vec{v}_j]$:
1. Execute $DA$;
2. $L_0 := max(L_0, L_j) + 1$;
   \[
   \forall p_k \in P: \vec{v}_i[p_k] := \begin{cases} 
   max(\vec{v}_i[p_k], \vec{v}_j[p_k] + 1) & \text{for } p_k = p_j, \\
   max(\vec{v}_i[p_k], \vec{v}_j[p_k]) & \text{otherwise};
   \end{cases}
   \]

$DA$ Delayed Addition, if ($isSent = true$) {
$\vec{v}_i[p_i] := \vec{v}_i[p_i] + 1$;
$isSent := false$;
}

1 We assume the entry of a replica $R_k$ is represented by $\vec{v}_i[p_k]$, where $p_k$ is the prime number of $R_k$ instead of an integer index.

---

**Fig. 1.** A case in which a replica is destroyed and created. The entry of the destroyed replica is deleted immediately. If version vectors are in the forms of anonymous arrays, there might be ambiguity to update $v_3^i$ and $v_1^i$.
Hence, \( LP_i \) should be a real number. For example, in Fig. 2, the last \( \log \delta \) value of \( R_2 \) becomes \([3, 8, \log_\delta 2^{3 \times 31}]\).

For two \( \log \delta \)’s \( \ell V_i = [p_i, L_i, LP_i] \) and \( \ell V_j = [p_j, L_j, LP_j] \), Definitions 1 and 2 are redefined as follows.

**Definition 3 (Dominant relation \((\ell V_i \prec \ell V_j)\)).** \( \ell V_j \) dominates \( \ell V_i \), i.e., \( \ell V_i \not\prec \ell V_j \), iff: (1) for \( p_i = p_j, L_i < L_j \), or (2) for \( p_i \neq p_j, L_i < L_j \) and a discriminant \( \Delta(\ell V_i, \ell V_j) = \beta^\delta \in \mathbb{N} \) for \( \delta = \log_\delta p_i \).

**Definition 4 (Conflict relation \((\ell V_i \perp \ell V_j)\)).** \( \ell V_i \) and \( \ell V_j \) are in conflict, i.e., \( \ell V_i \perp \ell V_j \), iff: neither \( \ell V_i \prec \ell V_j \) nor \( \ell V_j \prec \ell V_i \).

In Definition 3, the discriminant is

\[
\Delta(\ell V_i, \ell V_j) = \beta^\delta = \prod_{p_k \in \mathbb{P}} p_k^{(\tilde{v}_j(p_k) - \tilde{v}_i(p_k))}.
\]

Clearly, if condition (2) of Definition 1 is satisfied, \( \Delta = \beta^\delta \) becomes a natural number. However, \( LP_i \) of an irrational number cannot be represented without errors in digital systems, and the discriminant might be misjudged as a result of these errors; note that the errors equally occur to all replicas because they are mathematically deterministic. Hence, the error needs to be exactly predicted and suppressed.

Let \( \delta \) be the value of a real number representation of a positive irrational number \( \delta \). We assume that the error between the two, i.e., \( \varepsilon = \delta - \tilde{\delta} \), is small enough to be close to zero. Then, the error of the discriminant is \( |\varepsilon| = |\beta^\delta - \beta^\tilde{\delta}| \), which could be expanded as follows.

For \( \tilde{\delta} = \min(\delta, \tilde{\delta}) \),

\[
|\varepsilon| = |\beta^\delta - \beta^\tilde{\delta}| = |\beta^\tilde{\delta} + |\varepsilon| - \beta^\delta| = \beta^\tilde{\delta} |\varepsilon| - 1 = \beta^\tilde{\delta} \left( |\varepsilon| \ln |\beta| \right) - 1.
\]

If \( e^{\varepsilon |\ln \beta|} \) is substituted by Maclaurin series, then

\[
|\varepsilon| = |\beta^\delta\left( |\varepsilon| \ln |\beta| + \frac{|\varepsilon| |\ln \beta|^2}{2!} + \frac{|\varepsilon| |\ln \beta|^3}{3!} + \cdots \right)|.
\]

From this infinite series, the following inequality for \( |\varepsilon| \) is obtained:

\[
\beta^\delta |\varepsilon| \ln |\beta| < \varepsilon |\beta^\delta\left( |\varepsilon| \ln |\beta| + |\varepsilon| |\ln \beta|^2 + \cdots \right)\]
\[
\Leftrightarrow \beta^\delta |\varepsilon| \ln |\beta| < \varepsilon < \beta^\delta \left( |\varepsilon| \ln |\beta| + |\varepsilon| |\ln \beta|^2 + \cdots \right).
\]

The right term, \( \frac{|\varepsilon| |\ln \beta|}{1 - |\varepsilon| |\ln \beta|} = \frac{1}{|\varepsilon| |\ln \beta|} \approx |\varepsilon| \ln |\beta| \), if \( |\varepsilon| \ln |\beta| \) is small enough. Hence,

\[
\beta^\delta |\varepsilon| \ln |\beta| < \varepsilon < \beta^\delta |\varepsilon| \ln |\beta|
\]
\[
\Leftrightarrow |\varepsilon| \approx \beta^\delta |\varepsilon| \ln |\beta| \quad \text{for } \tilde{\delta} = \min(\delta, \tilde{\delta}).
\]

From formula (1), we can say that the following two conditions can reduce the error in the discriminant.

1. \( |\varepsilon| = |\delta - \tilde{\delta}| \) must be small.
2. Since \(|\varepsilon| \) is proportional to \( \ln |\beta| \), smaller \( \beta \) is better.
Based on these conditions, we make the following three suggestions for implementing $\log'$:

S1. Dividing $LP_1$ into integer and decimal parts.
S2. Fixing the precision of the decimal part.
S3. Letting $\beta = 2$.

Suggestions S1 and S2 make it possible to predict the error of $\delta$ and ensure that every $LP_1$ has some error bound. We implement the decimal part of $LP_1$ using $p$-bit raw array, i.e., with $p$ precision. The most significant bit of the array denotes $2^{-1}$, and the least $p$th one denotes $2^{-p}$. According to [18], since the rounding error from the decimal part is less than or equal to $2^{-p}$, the error of $\delta$ (say $\epsilon$), i.e., the subtraction of three terms, becomes $|\epsilon| \leq 3 \times 2^{-p}$.

By letting $\beta = 2$ (S3), S1 and S2 also facilitate the calculation of the discriminant. In general, $\beta^\delta$ for $\delta \in \mathbb{R}_+$ is calculated by using the relationship $\beta^\delta = e^{\delta \ln \beta}$ [19]. If $\beta = 2$ and $\delta = \delta_1 + \delta_f$, of which $\delta_1$ and $\delta_f$ are the integer and decimal part, respectively, then the discriminant $\Delta = 2^{\delta} = 2^{\delta_1 + \delta_f} = 2^{\ln \beta} \times e^{\delta_f \ln 2}$. After calculating $e^{\delta_f \ln 2}$, the multiplication of $2^{\delta_1}$ can be emulated by shifting $e^{\delta_f \ln 2}$ to the left by $\delta_1$ bits. This ensures better performance in determining the discriminant.

The error in $e^{\delta_f \ln 2}$ is also predictable. Assume $\tilde{\delta_f}$ and $\ln 2$ are the decimal representations of $\delta_f$ and $\ln 2 = 0.6931471 \ldots$, respectively. Letting $\epsilon_1$ and $\epsilon_2$ be the rounding errors of $\delta_f$ and $\ln 2$, respectively, then $|\epsilon_1| \leq 3 \times 2^{-p}$ and $|\epsilon_2| \leq 2^{-p-1}$ as stated above. Using these, $y = \delta_f \times \ln 2$ can be written as follows.

$y = \delta_f \times \ln 2 = (\tilde{\delta_f} + \epsilon_1)(\ln 2 + \epsilon_2)$

$= \tilde{y}_1 \ln 2 + \epsilon_1 \ln 2 + \tilde{y}_2 \epsilon_f + \epsilon_1 \epsilon_2$.

If $\tilde{\delta_f}$ and $\ln 2$ are used to calculate $y$, the error of $\tilde{y} = \delta_f \times \ln 2$ would be $\epsilon_3 = \epsilon_2 \tilde{\delta_f} + \epsilon_1 \ln 2 + \epsilon_1 \epsilon_2$. Due to $0 \leq \delta_f < 1$, $|\epsilon_1\epsilon_2| < |\epsilon_2\tilde{\delta_f}| < |\epsilon_1\ln 2|$. Hence, $|\epsilon_3|$ is dominated by the term $\epsilon_1\ln 2$ and bounded as $|\epsilon_3| \leq 3 \times 2^{-p-1} \ln 2 < 2^{-p+1}$.

From condition (1), the error bound of $e^{\delta_f \ln 2}$ is

$|E| \approx e^{\delta_1 \ln 2}|\epsilon_3| \approx 2^{\delta_1}/|\epsilon_3|$.

$< 2^{\delta_1} \times 2^{-p+1} < 2^{-p+2}$ ($: 0 \leq \delta_f < 1$).

Hence, errors can appear in the two least significant bits.

the bits that might have errors

1. $b_1b_2 \ldots b_{p-1}b_{p-2}$ $b_{p-3}b_{p-2}$ $b_{p-1}b_p$.

only two bits are contaminated, while the fore bits from $b_1$ to $b_{p-3}$ are trustworthy. Shifting the decimal point to the right by $\delta_1$ bits produces the value corresponding to the discriminant $\Delta = 2^\beta$. However, due to the error of the last two bits, the discriminant can be misjudged.

If $LV_i \not< LV_j$, $\Delta(LV_i, LV_j) \in \mathbb{N}$; that is, all the bits of the decimal part of $\Delta$ must be zeros. Due to the error, however, the trustworthy bits of $\Delta$ are all zeros if the error is positive, while they are all ones if the error is negative. To make up the error, we round $e^{\delta_f \ln 2}$ to the nearest on the last trustworthy bit $b_{p-2}$ so that all the bits of the decimal part can be zeros [18]. This ensures that the discriminant of the dominant relation is always determined correctly.

If $LV_i \not< LV_j$, $\Delta(LV_i, LV_j) \notin \mathbb{N}$; this means that the bits after the decimal point should be neither all zeros nor all ones. However, if the value of the decimal part of $\Delta$ is too close to either zero or one, the trustworthy bits might be all zeros or all ones, respectively. This causes a conflict relation to be identified incorrectly as a dominant relation.

As $\Delta$ increases, the trustworthy bits of the discriminant decrease, thereby causing the accuracy of the determination to decline. If $\delta \geq p - 1$, the discriminant determination is unavailable because no trustworthy bits remain. For example, assuming that $p = 64$ and $58 < h < 59$, only four bits are trustworthy: $64 - 2$ (the bits where an error appears) $- 58$ ($\delta_1$, i.e., the bits that must be shifted to the left) $= 4$. In this case, only when the decimal part of the discriminant for a conflict relation has a value in interval $A = [\frac{1}{15}, \frac{1}{16}]$, as shown in Fig. 3, can the discriminant be correctly determined. If it is in interval $B = (0, \frac{1}{15}) \cup (\frac{15}{16}, 1)$, the four trustworthy bits would be either ‘0.0000’ or ‘0.1111’ in binary, resulting in an incorrect determination that the discriminant of the conflict relation is misconceived as a natural number after being rounded on the last trustworthy bit. Nevertheless, we regard this relation as a dominant one because dominant relations frequently happen when two $\log'$s are a long way off.

To this end, $\log'$s have the following property.

$V_i < V_j \Rightarrow LV_i < \lhd LV_j$.

$V_i \not< V_j \Leftarrow LV_i \not< \lhd LV_j$.

This property is the same as that in the plausible clocks introduced by Torres-Rojas and Ahamad [16], which likewise sacrifice accuracy in detecting causality so as to achieve constant size vectors. Although $\log'$ is superior to typical plausible clocks [16] in terms of both accuracy and logging overhead, this paper does not compare $\log'$ with them because $\log'$s are obtained by the medium of prime version vectors and inherently not designed to be vector clocks but a concise logging form of version vectors. In-
instead, with respect to various system models, this paper focuses on showing the characteristics of log' errors in the following section.

4. Simulations

We implement log' using the MPFR library, which supports a multi-precision floating-point calculation [20]. The synchronization of replicas is simulated to examine the detection accuracy of log's. The simulations are designed as shown in Fig. 4, in which four parameters are defined as below:

1. \( R_1, \ldots, R_r \): \( r \) replicas,
2. \( T_1, \ldots, T_t \): \( t \) turns, at each of which a replica issues either an internal or a sync operation, or receives an rsync operation,
3. \( LV_1, \ldots, LV_{rt} \): log's of operations \( O_1, \ldots, O_{rt} \) listed in the order as shown in Fig. 4,
4. A sync operation is sent to another random replica within an arbitrary delay (< 10 turns). If a replica is a master, its sync operation is broadcast to all the other replicas.

To show the detection accuracy of log', we present accuracy graphs wherein every relation from a pair of distinct log's is compared with its counterpart of conventional version vectors. In Fig. 5, the x-axis represents a set of \( x \) log's, and the y-axis shows the ratios of conflict relations over all possible \( \binom{x}{2} \) pairs, depicted with the dashed line graphs of legend (c). As evident in the graphs of Fig. 5, the ratio of conflict relations declines as the set size grows. The solid-line graphs of legend (e) represent the error ratios at which relations are incorrectly detected, compared to the detection using conventional version vectors. Note that with log's, dominant relations are always detected correctly, but conflict relations might not be. Hence, the vertical difference between the two
classes of graphs reveals the ratio of the conflict relations being detected correctly. Overall, there are sets of log's, named error-free log' sets, of a certain extent (600–4800 log's) that contain no error despite very high ratios of conflict relations.

Fig. 5(a) shows the accuracy graph with regard to synchronization policies. In many real-world replication systems, replicas synchronize themselves mutually in order to make all replicas consistent. For example, Coda allows several master replicas to broadcast every sync operation [2], and in most collaborative editing systems, every replica broadcasts sync operations for responsiveness [5]. To this end, four synchronization policies are devised with 25 replicas: (p2p) every replica synchronizes itself with only one of other replicas, (m=5 and 10) 5 and 10 masters of total 25 replicas broadcast their sync operations, and (m=full) all the replicas broadcast their sync operations. Synchronization makes log's of different replicas have similar values, for which conflict relations tend to be detected correctly as stated in Section 3; thus, as the number of masters increases, accuracy improves. The rest of our simulations are done on m=full.

Obviously, the precision of the decimal parts affects accuracy, as shown in Fig. 5(b). The accuracy is rapidly enhanced as more bits are used for the decimal parts. For example, for 10000 log's, the errors are about 9% for p=256 bits, but less than 2% for p=512 bits. This result is evident because those log's that are generated at around the same time have a tendency to be in conflict relation, and the error-free region, where all detections are correct, is widened as more bits are used for decimal parts. Fig. 6 shows the region where errors appear for all the pairs of 10000 log's; such errors are highlighted with different colors. For instance, if some 4000th log is in conflict relation with some 0th or 8000th log's, the relations are correctly detected if 512 bits are used, but might not be for 320 bits.

Fig. 5(c) evaluates the effects of the replica size (from 25 to 100 replicas) on accuracy. To fix the number of log's, we consider only 5000 log's of the first 25 replicas under the condition that all the replicas have actively issued log's. According to the definition of the conflict relation, before two operations are generated at different replicas, if only one of the replicas had received some other operations from other replicas, the version vectors of the two operations must be in conflict relation. For that reason, as the replica size becomes larger, the conflict ratio rises, as shown in Fig. 5(c). Meanwhile, for r=25, the error-free set has about 1800 log's, but the set sizes are kept up around 1000–1200 log's for r=50–100. However, as the replica size becomes larger, the error ratio rises, since more operations out of the error-free set are in conflict relation.

Finally, we simulate the effects of membership changes. Amongst r=20 replicas, the replica of the smallest prime number retires every 6 turns, which means the retired replica neither issues nor receives operations any more. The retired replica is replaced with a new one that is assigned a new prime number and a new prime version vector. During 500 turns, a total of 80 replicas newly participate in the replication system, but only 20 replicas are at the same time in the active status that allows a replica to issue and receive operations. If conventional version vectors are used, they will have a hundred entries in the end.

Fig. 5(d) is the accuracy graph, in which 'static' describes the replication with no membership change while 'dynamic' is the result of the replication designed above. For log's of 128-bit and 320-bit precisions, their conflict ratios are similar, but the error ratios of dynamic replications are slightly higher than those of static repli-
5. Conclusions

We propose the $log'$ version vector and its implementation. $log'$ is obtained from its operating form, prime version vector designed to enhance the detection accuracy of $log'$. The primary virtue of $log'$ is that it requires no vector pruning. Hence, $log'$ is suitable to dynamic replication of limited membership. The simulation studies show that $log'$s are logged concisely with few errors in the full synchronization replication.

Errors in detecting conflicts with $log'$s equally appear to all replicas because they are mathematically deterministic. Therefore, despite these errors, $log'$s can offer a reasonable compromise for maintaining consistency among replicas. Indeed, an operation’s being dominant over some others can be interpreted that it has directly or indirectly experienced them. Respecting experience, in general, the effect of dominant operation is preferentially applied to replicas while reconciled effects are taken for conflicting operations. Even if two $log'$s of conflict relation are misjudged as a dominant relation, the dominant $log'$ must have gained much more experience than the other. In this regard, incorrect detection with $log'$s is acceptable enough for some replication systems to adopt $log'$s.

References


