

Bandwidth Efficient Web Object Change Interval Estimation

Clancy Malcolm, Grenville Armitage
Centre for Advanced Internet Architectures
Swinburne University of Technology
Melbourne, Australia
{cmalcolm,garmitage}@swin.edu.au

Abstract – Estimating the average time between changes to objects on the Web has many applications including content distribution networks, search engines, web site monitoring and measuring web dynamics. This paper provides formulas for estimating average object change interval and efficient visit scheduling algorithms. We show that the models developed are more accurate and produce less server and network load than previously devised techniques.

I. INTRODUCTION

This paper contributes efficient methods for a robot on the Web to estimate the average change interval of objects. We developed these techniques for use in reasearch on the relationships between an object's change interval and its ability to be cached in a content distribution network, but they are also applicable to other areas such as search engines, remote file backup systems and data warehouses. For example, [1] and [2] show that if a search engine is able to accurately estimate an object's average change interval then it can keep more of its index up to date.

By improving on previously studied formulas and devising efficient visit scheduling algorithms, we have developed models that on average use less bandwidth and are more accurate than previously published models. No previous estimates of average object change interval are assumed and the model handles objects that do not report the date of last change.

When estimating a web object's average change interval, a number of constraints are involved:

- It is near impossible to obtain a complete change history of an object
- Many web servers do not provide the time of last change.
- Sampling objects too frequently places unacceptable load on servers and Internet links
- It may be impractical to sample over a long period of time such as many years

Our models aim to achieve accurate results within these constraints while minimising network traffic.

Table 1 introduces the terminology used in this paper.

<i>Term</i>	<i>Definition</i>
Web object	Any item that can be accessed using HTTP
Visit	A HTTP request to retrieve a web object
Visit scheduling	The process of determining the next time that an object should be visited
Change	Any variation in an object's response body – we ignore changes in response headers as some of these (e.g. Date header) are designed to change for every response
Detected change	A change that is observed by the robot visiting the object before the next change
Undetected change	A change that is not observed due to the object changing again before the next visit
Average object change interval	The average time between an object's changes. (An object's <i>rate of change</i> is the inverse of its average change interval)
Estimated average change interval	Average change interval calculated by the robot after visiting an object
Visit interval	The time between two visits (for example, 1 day)
Sampling period	The time between the first visit and the last visit (for example, 4 months)
Last-modified date	Data in the HTTP response headers that indicate the last time an object changed (not provided for all objects)
Object age	The amount of time since an object last changed

Table 1. Terminology

II. RELATED WORK

Web robot scheduling has been explored in various papers. [1] discusses robot scheduling policies that minimize a cost function based on the fraction of time that pages in an index are out-of-date. [2] focuses on optimal robot scheduling in search engines based on freshness and age metrics. Our study differs from these papers in that our focus is estimating change intervals.

Formulas for estimating frequency of change (the inverse of the average change interval) were developed in [3]. This paper extends those formulas by making them less susceptible to irregular visit intervals and more accurate when estimating average change intervals. [4] provides alternative extensions to the formulas in [3], but these

formulas require very intensive computations and their accuracy is unknown.

Scheduling based on sampling of related documents is discussed in [5], but for our studies we assumed that the objects of interest were a broad range of unrelated URLs where such a method cannot be used.

III. METHOD

The formulas and scheduling algorithms covered in this paper were tested using a web object simulation [6] to allow the comparison of actual (simulated) object change intervals to those estimated under various conditions. Many papers including [2] and [3] have shown that web object changes can be accurately modeled as a Poisson process and hence the simulator modeled the time between changes using a negative-exponential model. (In a Poisson process the time between events can be modeled using a negative-exponential distribution.) The simulator was able to simulate documents both with and without last-modified date.

To evaluate our models we examine six scenarios, as detailed in Table 2. Scenarios 1 and 3 use a fixed visit interval with the estimation formulas suggested in [3]. We then compare these results to those obtained using our formulas (Scenarios 2, 3, 5 and 6) and our scheduling algorithms (Scenarios 3 and 6).

	<i>Last-Modified Provided?</i>	<i>Estimation Formula</i>	<i>Visit Interval</i>
<i>Scenario 1</i>	Yes	See [3]	24 hours
<i>Scenario 2</i>	Yes	See Section V	24 hours
<i>Scenario 3</i>	Yes	See Section V	Variable
<i>Scenario 4</i>	No	See [3]	24 hours
<i>Scenario 5</i>	No	See Section VI	24 hours
<i>Scenario 6</i>	No	See Section VI	Variable

Table 2. Simulated scenarios

We assess the accuracy and efficiency of our models using three metrics:

- **Bias** – The ratio of the mean estimate and the actual value of an object's average change interval. Ideally this ratio is 1 (i.e. the mean estimate and the actual value are the same), making the estimate unbiased.
- **Standard Deviation** – A measure of the typical difference between individual estimates and the mean estimate. Ideally the standard deviation is very low indicating that most estimates are very close to the mean estimate.
- **Visits per Object** – The number of visits required to make the estimate. Fewer visits will generally result in lower network and server load.

IV. RESULTS

Figures 1a, 1b and 1c illustrate the bias, standard deviation and visits per object for each of the six scenarios described in Table 2. Where two different

scenarios produced similar results there plots have been combined for clarity.

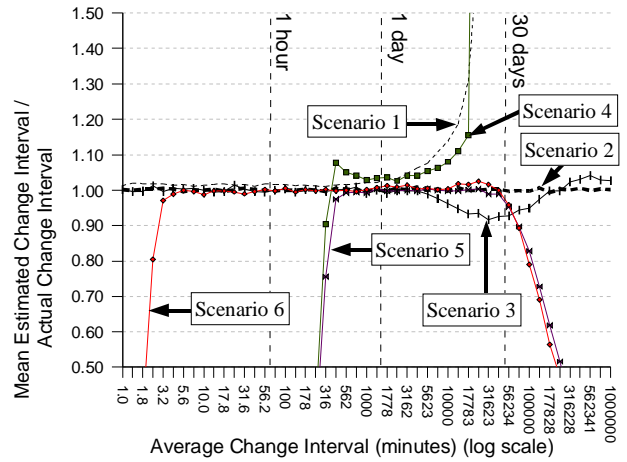


Figure 1a. Ratios of mean estimates to actual average change intervals to illustrate bias. Note the small range used on the Y-axis to add clarity.

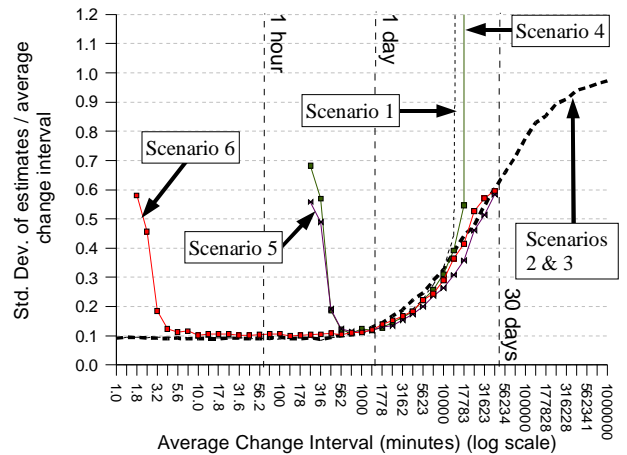


Figure 1b. Standard deviation of estimates

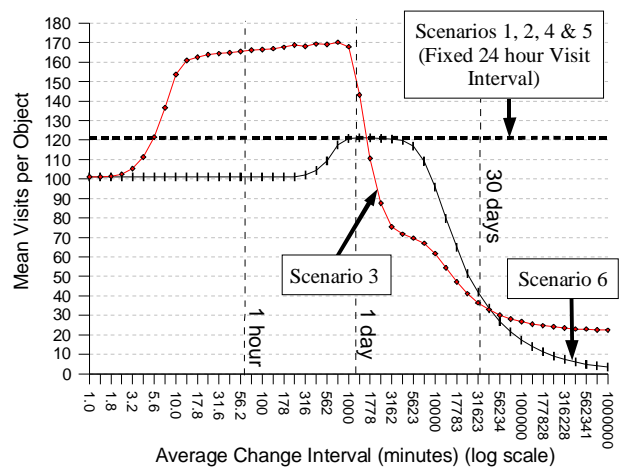


Figure 1c. Mean visits per object

The results show that compared to previous work (scenarios 1 and 4) the formulas and scheduling algorithms we have developed generally achieve results with low bias and standard deviation while making fewer visits than using a fixed visit interval. In the following two sections we examine the formulas and scheduling algorithms that we have developed and discuss these results in more detail.

V. ESTIMATING OBJECT LIFETIME WITH LAST-MODIFIED DATE

An intuitive estimator of an object's average change interval can be defined as:

$$\text{avgChangeInterval} = \frac{\text{samplingPeriod}}{\text{changes}} \quad (1)$$

However, this formula produces a large bias unless the visit interval is much shorter than the average change interval as it does not account for undetected changes (see [3]).

In a Poisson process the expected time to the previous event is equal to the expected time between events. Hence a more accurate estimator of an object's average change interval is to use the mean of the object ages over all visits:

$$\text{avgChgInterval} = \frac{\text{totalAges}}{\text{visits}} \quad (2)$$

The age of an object is the difference between the last-modified date and the date that it was retrieved. *totalAges* is the sum of the ages over all visits. *visits* is the number of times the object was visited.

[3] states that using this approach for estimating an object's *rate* of change introduces bias when the visit interval is much smaller than the object's average change interval. Our simulations indicated that the bias was caused by the inversion of the average change interval to calculate rate of change. When the estimates are not inverted they provide an unbiased estimator of an object's average change interval (providing that a constant visit interval is used).

The problem with a constant visit interval is that visiting an object far more frequently than it changes does not increase the accuracy of the results and hence creates unnecessary network and server load. If we had an estimate of an object's average change interval before we started sampling then we could use this to select an optimal visit interval, but often this information is not available.

The general approach used to overcome these problems was to construct an estimate of an object's average change interval after each visit based on all the visits to-date and then use this estimate to determine the appropriate timing for the next visit.

Listing 1 shows the visit scheduling algorithm we developed for objects with a last-modified date. The algorithm does not require a prior estimate of average change interval and avoids unnecessarily frequent visits.

After each visit *chgIntervalEst* is set to the current estimate of the object's average change interval and *numChanges* is set to the number of detected changes. Next the *getNextInterval* function is called and it returns the amount of time to wait before visiting an object again. The function can also return -1 to indicate that the object should not be visited again.

```
function getNextInterval()
{
  if (numChanges >= MAX_CHANGES)
  {
    return -1;
  }
  interval = TARGET_INTERVAL;
  if (interval < chgIntervalEst/MAX_RATIO)
  {
    interval = chgIntervalEst/MAX_RATIO;
  }
  return interval;
}
```

Listing 1. Algorithm to calculate visit interval for objects with a last-modified date.

The all upper-case identifiers are constant parameters. The results shown in Figure 1 were obtained using the parameter values shown below in Table 3.

MAX_CHANGES	100
TARGET_INTERVAL	1 day
MAX_RATIO	5

Table 3 – Example parameters for scheduling visits to objects with a last-modified date

Using this scheduling algorithm did introduce a small amount of bias (between 1% and 10%) for objects with an average change interval of more than 7 days due to variance in the access intervals. However, Figure 1c shows that this algorithm substantially reduces the number of visits to objects with a long average change interval compared to a fixed 24 hour visit interval. The standard deviation observed was very similar to a constant 24 hour visit interval.

As shown in figure 1b, The standard deviation of the estimates grows quite large for objects with a relatively long average change interval, regardless of the visit scheduling used. For example, the standard deviation is 39% for objects with an average change interval of 10 days. This variance is due to the relatively few changes that can be observed during the sampling period – an object with an average change interval of 10 days will only change an average of 12 times during a 120 day sampling period and the intervals between these changes can vary significantly from the average change interval, making the variance in the results unavoidable. The affect of the high standard deviation is that individual estimates cannot be considered accurate, but as the results are unbiased, trends for a large group of objects can be identified.

VI. ESTIMATING OBJECT LIFETIME WITHOUT LAST-MODIFIED DATE

When the last-modified value was not provided for an object (or could not be considered accurate) we could not

calculate an object's age with sufficient accuracy to apply the estimator previously discussed.

The estimator presented in [3] performs poorly when using a variable visit interval and using a fixed visit interval is inefficient unless we already have an estimated average change interval.

[4] presents an alternative estimator that does not assume fixed visit intervals, but this estimator cannot be quickly computed for a large number of documents and as such it is not scalable.

We have modified the estimator in [3] to reduce the bias caused by irregular access. Our new estimator is shown below.

$$\begin{aligned}
 avgDetectionInterval &= \frac{samplingPeriod - totalUnchangedTime}{changes} \\
 avgChgInterval' &= \frac{avgDetectionInterval}{\log\left(\frac{samplingPeriod}{totalUnchangedTime}\right)} \\
 avgChgInterval &= \frac{avgChgInterval'}{1 + \frac{avgChgInterval'}{samplingPeriod}} \quad (3)
 \end{aligned}$$

totalUnchangedTime is the total of the visit intervals during which the object does not change. *avgChangeInterval'* is an initial estimate of the average change interval that exhibits bias as the estimate approaches the sampling period due to the distribution of the estimates. *avgChangeInterval'* can be inverted to obtain an unbiased estimate of the average rate of change. The final formula corrects the bias in *avgChangeInterval'* for estimating the average change interval.

This estimator is most accurate when the visit interval is close to the object's average change interval, so we base our scheduling algorithm on a function that uses an estimate of the object's average change interval from previous visits as the next visit interval. However, this introduces a number of issues that must also be addressed in our scheduling algorithm:

- We need a lower limit on the visit interval so that we do not overload the network or the server by using very short visit intervals
- We need to stop sampling once a certain number of changes have been detected – otherwise objects that have a very short average change interval will consume excessive resources
- Some objects should be visited more frequently to ensure that a sufficiently large number of visits are made during the sampling period. For example, if an object has an average change interval of 3 days and we were using a sampling period of 120 days then visiting it once every 2 days provides additional accuracy over visiting it once every 3 days.

Listing 2 shows the algorithm developed for use where the last-modified date is not available. The function is

called in the same way as the function used where the last-modified date is available.

```

function getNextInterval()
{
  if (numChanges >= MAX_CHANGES)
  {
    return -1;
  }

  interval = chgIntervalEst;

  if (interval > SOFT_MAX_INTERVAL)
  {
    interval = SOFT_MAX_INTERVAL;

    if (interval <
        chgIntervalEst/MAX_RATIO)
    {
      interval = chgIntervalEst/MAX_RATIO;
    }
  }

  if (interval <= MIN_INTERVAL)
  {
    return MIN_INTERVAL;
  }
  else
  {
    return interval;
  }
}

```

Listing 2. Algorithm to calculate visit interval for objects without a last-modified date.

Table 4 shows a set of parameters for use with a 120 day sampling period that are able to match the accuracy of a fixed visit interval, but over a greater range of average change intervals and making less visits per document in many cases.

MAX_CHANGES	100
SOFT_MAX_INTERVAL	2 days
MAX_RATIO	3
MIN_INTERVAL	10 minutes

Table 4. Example parameters for scheduling visits to objects without a last-modified date

Figure 1 shows that compared to visiting objects once every 24 hours (scenario 5), the adaptive scheduling algorithm (scenario 6) greatly increases the range of change intervals where bias and standard deviation are acceptably low. This algorithm produces reasonably accurate results for objects with an average change interval of half the minimum visit interval. For example, objects with an average change interval of 5 minutes produce estimates that are 0.85% biased and have a standard deviation of 11.7%. This is a definite improvement over a fixed 24 hour visit interval which only achieves bias of less than 1% when the average change interval exceeds approximately 9 hours.

The estimates also become biased as the object's average change interval approaches the sampling period. For example, the estimates are more than 5% bias in scenario 6 when the average change interval is more than 40 days. This is because without a last-modified date it is not

possible to make estimates that are larger than the sampling period and hence as the probability of an object remaining unchanged for the duration of the sampling period increases, the bias also increases. This highlights the need to use a sampling period of at least three times the object's average change interval (or longer still if low standard deviation is also important).

VII. NETWORK AND SERVER LOAD

It is important to consider the impact of our visit scheduling algorithms on network and server loads. Many previous studies have visited objects once per day. Figure 1c compares the visits per object for our scheduling algorithms to a fixed 24 hour visit interval. It can be seen that in most cases the adaptive visit scheduling algorithms produce less visits per object. In [7] it is estimated that the average age of an object on the web is 62 days (approximately 90,000 minutes) – at this level both the adaptive visit scheduling algorithms result in a significant bandwidth saving of at least 75% compared to daily visit scheduling.

One drawback of using the adaptive visit interval when the last-modified date is not available is that the average visit interval is short at the start of the sampling period and this may cause additional network and server load during this time. However, in most cases the overall network and server load will be less as longer visit intervals are used later in the sampling period.

VIII. FUTURE WORK

The algorithms described in this paper have not been verified with real data – they rely on simulations of a Poisson process which has been shown by others to be a reasonable model for object changes on the Internet. Practical verification of these algorithms is important, but will take months and many gigabytes of bandwidth to complete. Furthermore it would be useful to analyse the performance of these algorithms for objects whose changes cannot be modeled by a Poisson process – for example objects that change at the same time each day.

HTTP/1.1 provides a number of features that can be used to reduce the network and server load. Two of the most useful are conditional requests, which allow a client to request a document if it has changed since it was last

retrieved, and content compression. These features are optional and hence are not supported for all objects, but they provide significant benefits when they can be used. Using these features to reduce network and server load is not explored in this paper, but the extent of their availability and usefulness to change interval estimation could be the subject of future work.

IX. CONCLUSION

Estimates of average object change interval exhibit a high variance when few changes occur during the sampling period. Hence it is critical to use a sampling period that is much longer than the change intervals being measured when results for individual objects are important. Trends in a large number of documents can be accurately identified, although the distribution of the results tends to be smoother than the true data due to high standard deviation of the estimates. These limitations are primarily due to the variance in input data and we believe that they are unavoidable.

When a suitably long sampling interval is used the formulas and scheduling algorithms we have developed can give results that are more accurate than those presented previously while placing significantly less load on the server and network links. These results can provide benefits to a range of applications including content distribution networks, search engines, web site monitoring and measuring web dynamics

REFERENCES

- [1] E. G. Coffman, Z. Liu, and R. R. Weber. *Optimal robot scheduling for web search engines*. Journal of Scheduling, 1997
- [2] Junghoo Cho and Hector Garcia-Molina. *Synchronizing a database to improve freshness*. Proc. of ACM SIGMOD, 2000
- [3] Junghoo Cho, Hector Garcia-Molina. *Estimating Frequency of Change*. Technical Report, February 2000
- [4] Laurent Mignet, Mihai Preda, Serge Abiteboul, S'ebastien Ailleret, Bernd Amann, and Am'elie Marian. *Acquiring XML pages for a WebHouse*. In proceedings of Base de Donn'ees Avanc'ees conference, 2000
- [5] J. Cho and A. Ntoulas. *Effective change detection using sampling (extended version)*. Technical report, UCLA Computer Science Department, 2002
- [6] Web Object Change Interval Estimation Simulator, <http://www.caia.swin.edu.au/ice/change-simulator.html>
- [7] B.E. Brewington and C. Cybenko. *Keeping up with the changing web* IEEE Computer, 33(5):52-58, May 2000