We designed four kinds of cache policies including Random, FIFO, Least Recently Used, and Largest Size First. We also tested the client latency of no caching. We tested our experiment in different conditions, especially different cache size, different kinds of workload and different requests number.

figure1.

Figure 1 shows the hit rate of cases where cache size is from 256KB to 2MB. We tested four different cache policies for 500 requests in uniform workload and the change cache size each time. As you can see, in uniform workload, the hit ratio of Random and FIFO is almost same and the URL might be a little bit better than these two policies. To our surprise, the MAXS is much better than other three policies. So we did some extra experiments to figure out why the MAXS performs so well later. Because in uniform workload, each URL has an equal chance to be called, in other word, the pages are non-relatively, so Random, LRU, and FIFO have similar hit ratio really makes sense.

Another thing which is showed in figure1 is that with the increasing of the cache size, the hit ratio becomes better and better. This is also in our expectation(符合我们的预期), because larger cache size means we can save more pages in one time, which increases the hit ratio.

figure2.

Figure2 is relative to the figure1, which shows the latency of different cache sizes. Normally higher ratio means the smaller latency because once we got a hit in cache we do not need to pull it from Internet again, which is also verified in figure2. The time latency of Random, LRU and FIFO is similar while the time latency of MAXS is much smaller than these three policies. And with the increase of the cache size, the time latency keeps decreasing.

In this part of experiment, we also test the time latency of the No Caching, as we expected, the time latency of the No Caching is much higher than any other policies. Also when the cache size is small enough, like 256KB, the time latency of using the cache policy is almost same as the No Caching.

figure3.

Next step of the experiment is testing the program in distrusted data.(就是说有lamad的情况下). We set the lamad equals 0.1. And the distribution of requests is like figure4. (figure4 就是展示一下lamad = 0.1的适合，url request是如何分布的。就可以直接放你之前展示workload分布的图的标号。)

In this situation, we expected the LRU would perform much better than any other policies. But to our surprise, the MAXS still beats any other policies and have a distinct leading. (向上文说的一样，之后我们会分析为什么MAXS有这么好的表现。) However, it's worth mentioning that with the increasing of the cache size, the leading of the MAXS turns smaller. Because in this situation, all of the policies could save more pages, the accuracy of the page match turns more important. (这里的意思是，随着Cache Size的变大，每一个policy都可以存储更多的page了，那么page hit的准确性变得更为重要了，所以MAXS领先幅度不那么大了。)

Actually, comparing to the uniform workload, the LRU has a better performance in this condition. LRU has about 5% hit ratio leading than the LRU and FIFO, which is also in our expectation. We believe if increase the size of URL List (now it is 50 URLS totally), the LRU would perform better than now it does. Because LRU might keeps the highest hit accuracy. (这里就说在URL变得更多之后，我们觉得以后LRU表现会更好，因为相比于其他的FIFO和Random他有更高的预测的准确度。)

figure5.

Figure5 is relative to the figure3, which shows the latency of different cache sizes. As we analyzed before, high hit ratio is along with small time latency and the No Caching has the worst time latency too.

figure6

figure7

In order to test the performance of our cache polices in a larger scale, we changed 500 times request to 1000 times requests. The results shows that no matter the hit ratio or the time latency, turning 500 times requests to 1000 times requests doesn’t change a lot. As we can see, in figure6 and figure7, the line between 500 and 1000 is almost horizontal.

figure8.

In order to understand how the distribution of the data would result the hit ratio and time latency, we tested all cache policies in 1MB cache size while the LamaD changes from 0.05 to 0.5. As the Lamad increases, the URL requests would concentrate. So when the lamad is 0.5, almost all the URL requests would occurs in front few URLS. (URL Request would 发生在前几个URL中), and hit ratio is certainly high. Actually, the hit ratio of LRU and MAXS is 96.4%, which means after LRU and MAXS adding the new pages to the cache at beginning; they cover the all the rest URL requests. In other word, we could say they do not have even one page fault. (这里说的是，LRU和MAXS的hit ratio是96.4%，除去最开始需要把新的page加入Cache之后，他们Cover了之后所有的URL请求。)

Also we can say, when the lamad is really small, like 0.05, the distribution of the data is similar to the uniform model. That’s why the gap between MAXS and other policies are bigger than other time when the lamad is 0.05.(这里说的是lamad很小的时候就像uniform一样了，然后MAXS的领先优势就较大。)

figure9

Figure9 is relative to figure8, which shows the client latency of four cache policies. As we discussed before, the small client latency stands for good performance. So figure9 verify the performance of these four cache policies from another aspect.

Figure10

Here is the webpage size of all our URLS. As we said before, to our surprise, the MAXS policy has a real good performance. So here we go a step further to why it would has such a great performance. First, we can observe that most of our web page is real small, while some others webpages might be 10 times bigger than smaller ones. So in the uniform model, MAXS policy might keep much more pages than other policies, for example, in 1MB cache size, MAXS might keep about 30 URLS in cache at one time, which is much better than Random, LRU, and FIFO. So it could explain why in the uniform model, MAXS has such good performance. Then how about the exponential mode? (第一节说的是在uniform的情况下，因为总的来说小的webpage比较多，那么MAXS存的会比较多。命中率自然高，举了一个1MB的例子。)

In figure10, represented in the x-axis, our URLS are sorted by the frequencies of the appearance in exponential model. (这里说的是X轴的URL是按照在exponential mode中出现的比例排列的。) As we can see, there are many small webpage size URLs in the first few URLs, only the twitter.com and youtube.com has big web pages. So in exponential model, the MAXS also could save a lot of high occurrence rate webpages. That’s why MAXS also performs good in exponential mode. (第二节说的是在exponential情况下，因为头几个的webpage size也都不大，所以MAXS还是可以比较好的表现。)