NSE Nsock Performance Review

Patrick Donnelly

6 July 2009

I. Background and Motivation The NSE Nsok binding is used to allow scripts parallel and efficient access to sockets. Thus far, we have allowed up to 10 (or –max-parallelism) scripts to have open sockets at any time. This is a necessary restriction that prevents Nmap from greedily using all the network resources on the host.

Over the last month, we have done testing on this library to explore the effect of this static maximum on concurrent sockets. In particular, we want to experiment with the effect raising parallelism has on the length of an NSE scan. In unrelated testing prior to this formal inquiry, I found that increasing the parallelism to 50, from 10, would decrease overall scan time of a specific test by 90%. After discussing this situation with Fyodor and David Fifield, we decided to more rigorously test the Nsock library binding to find ideal values of parallelism and the effect raising parallelism has on a specific scan.

II. Tests Structure First, these tests, as most do, require a static input. We use 100 static IP addresses of common web servers so each scan is more likely to encounter the same conditions. Due to firewalls and other varying circumstances, we cannot always guarantee the same conditions, of course. In the tests, usually about 5 hosts would change port states between two sample scans.

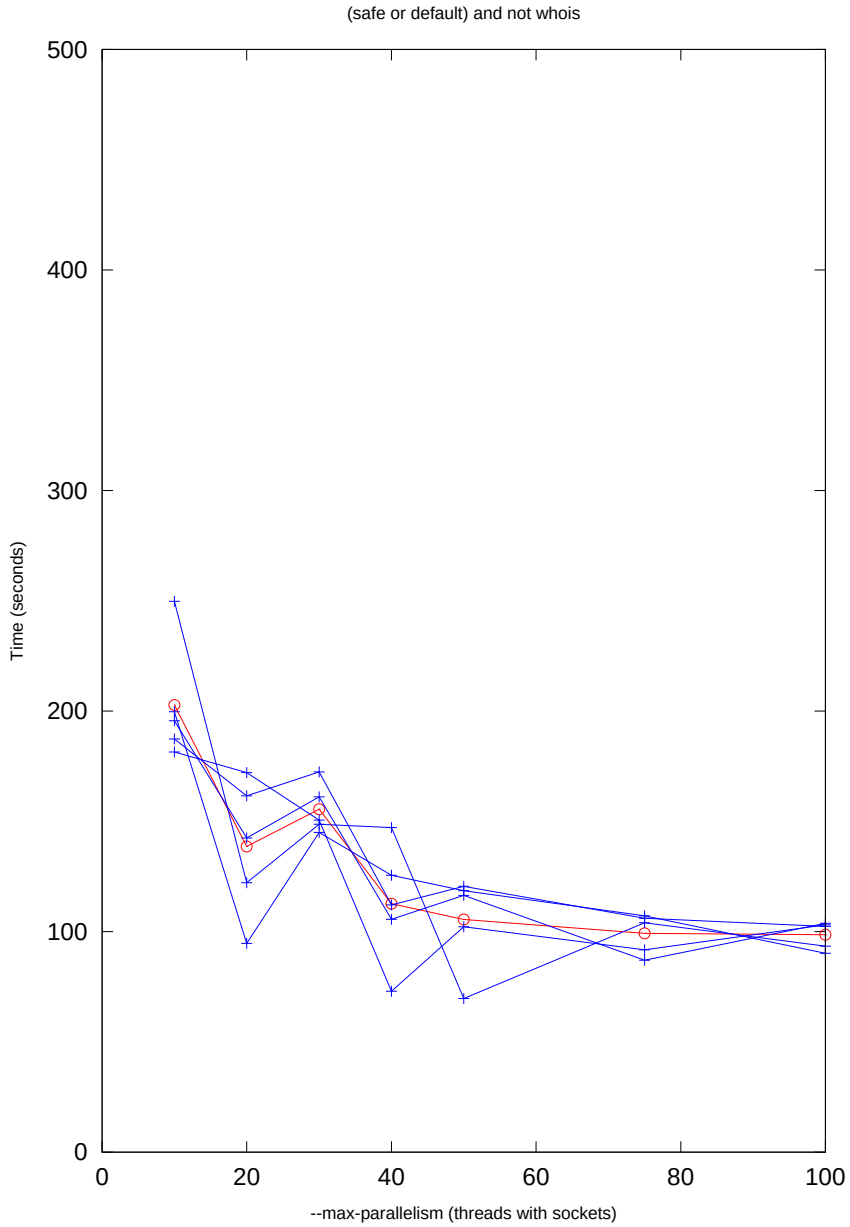
For our first scan group, I used the following example invocation in the Nmap trunk:

```
./nmap -v -d --datadir . -n -PN --host-timeout 10m --min-hostgroup 512 --max-
    parallelism $i -p 20-25,53,80,443,113,8080,110,995 --script "(safe or
    default) and not whois" -iL ~/common_ips.txt -oX safe_or_default_${i}_${k
    }.xml -oN safe_or_default_${i}_${k}.out 2>&1 | tee safe_or_default_${i}_${
    k}.terminal
```

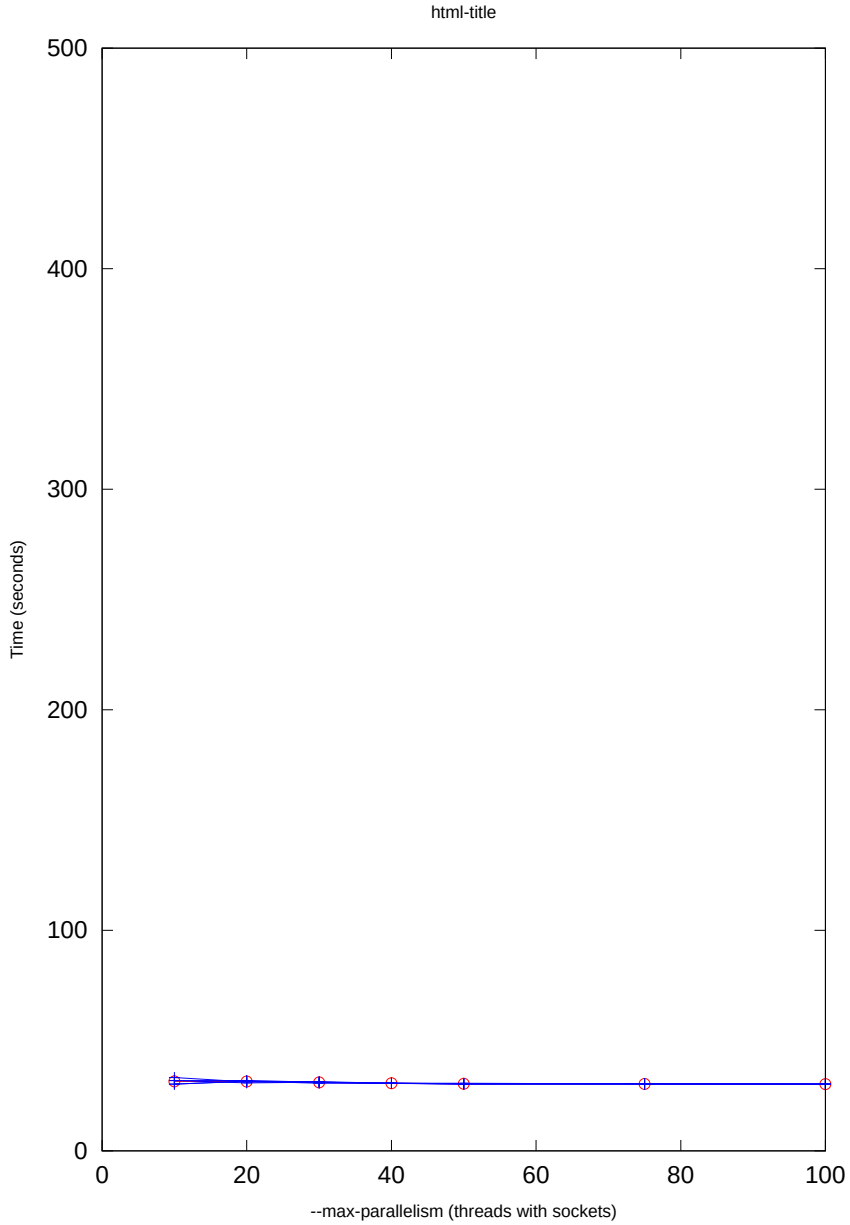
We test each parallelism value (i) 5 times (k). Similarly, we ran tests using other script scans: "html-title" and "not intrusive".

Each scan will obtain an average from 5 samples for each parallelism value. We use the following parallelism values: 10, 20, 30, 40, 50, 75, 100. For each script scan, these values get a good idea of the effect parallelism has on a different number and variety of scripts.

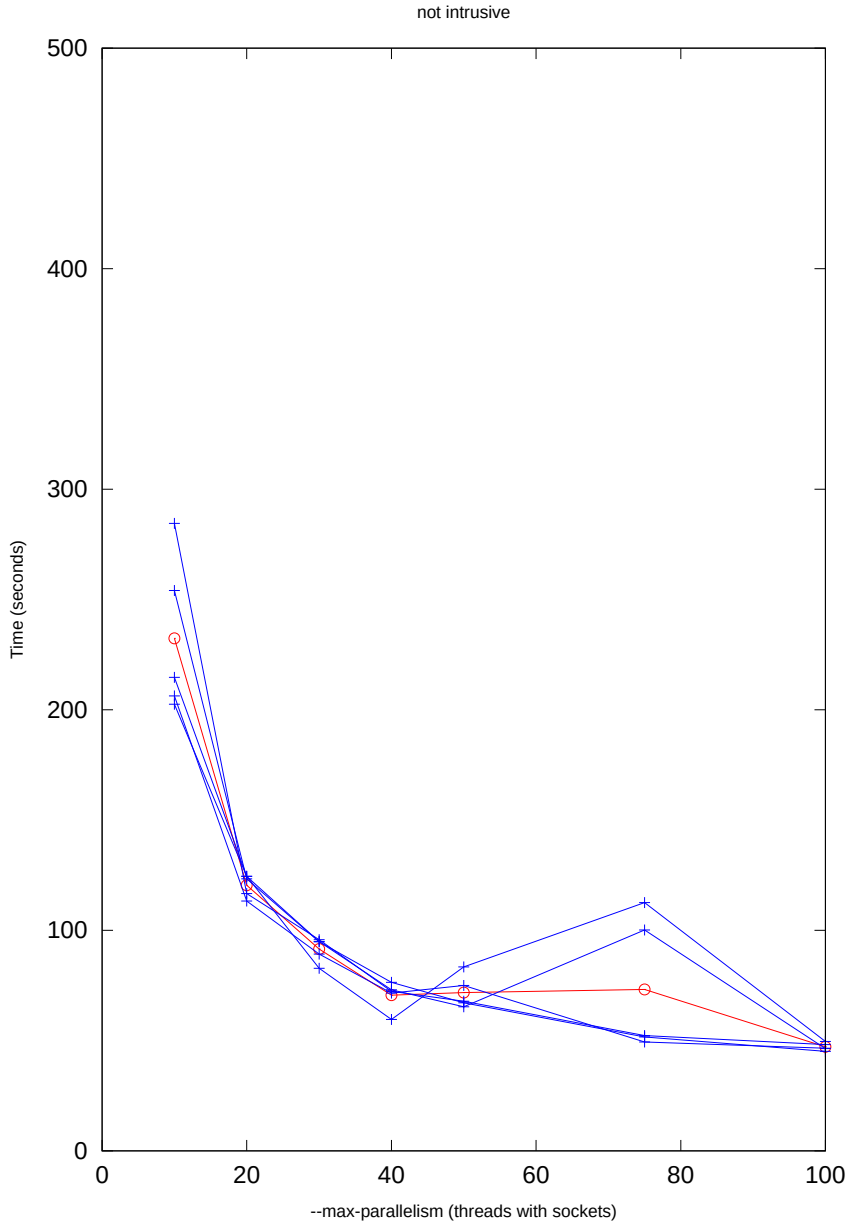
III. Results Here are results from the Nmap trunk. Each blue line represents a sample while the red line indicates the average of those samples.



–script "(safe or default) and not whois" This graph shows a general exponential decay regression as parallelism increases. With this scan type, we have 29 different scripts running with approximately 750 threads running on average. You can notice that around 30 to 40 –max-parallelism we reach a "sweet spot" where we get fewer benefits as parallelism increases further.

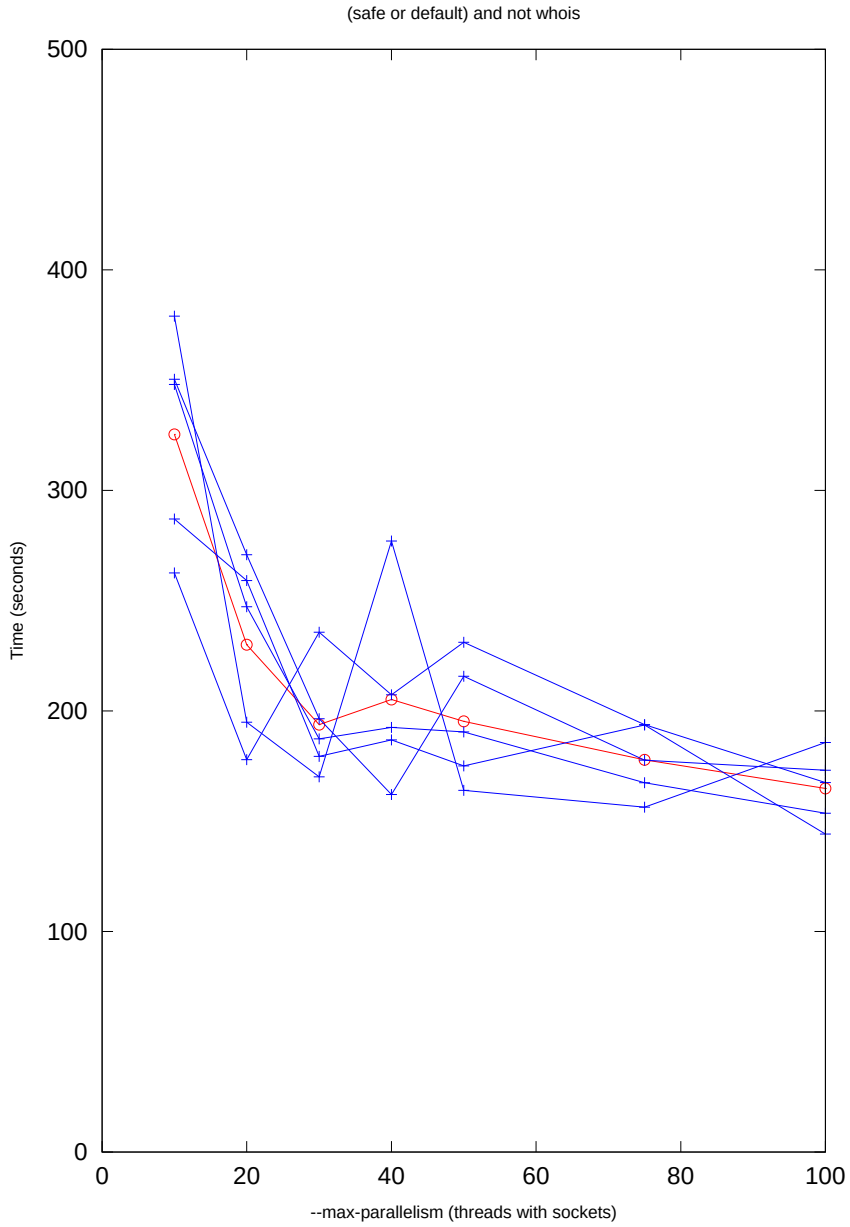


–**script "html-title"** This graph is useful for demonstrating a lack of any real benefits for a single script that runs against each host on port 80. On the other hand, it is good to note that performance does not degrade either.

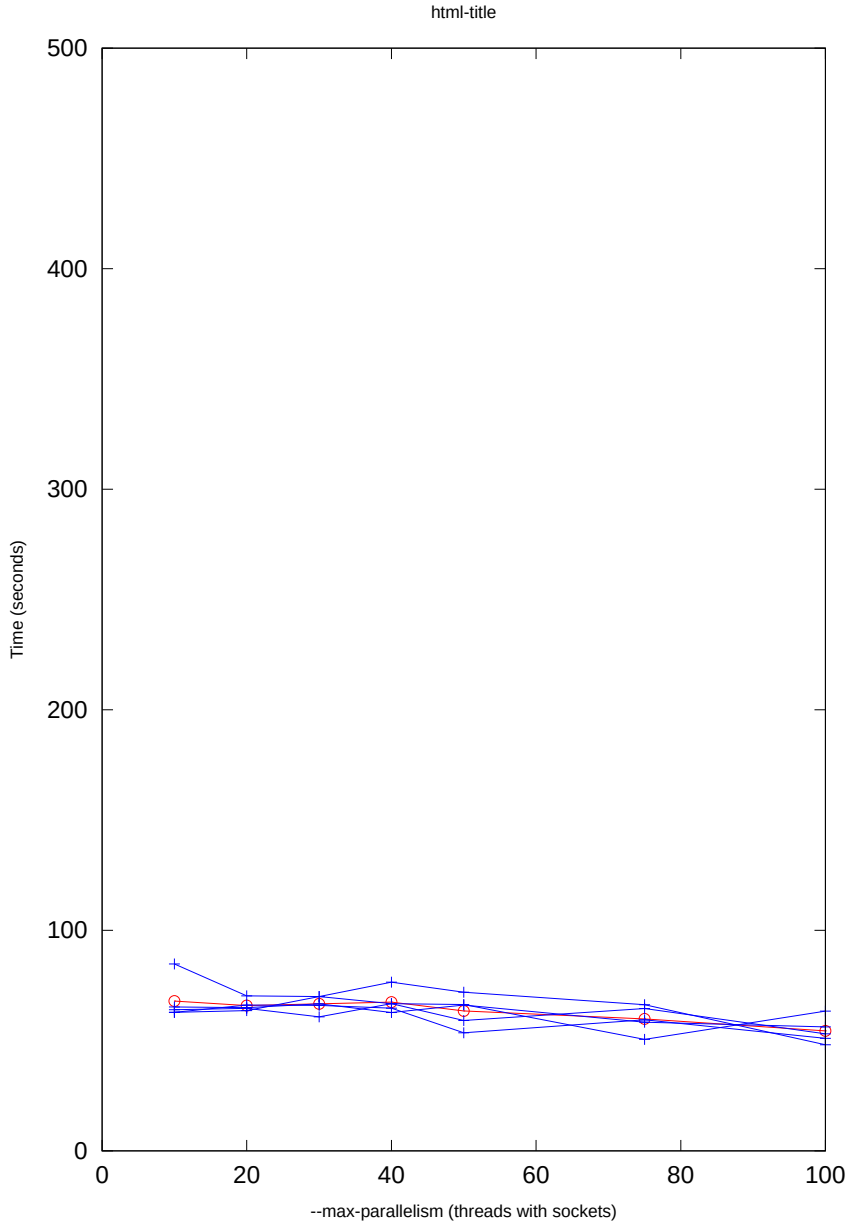


–**script "not intrusive"** We see here that, again, the graph takes on an exponential decay as parallelism increases. Here we have 32 scripts running with approximately 925 different threads. Again, around 40 parallelism we get less benefits from increased parallelism.

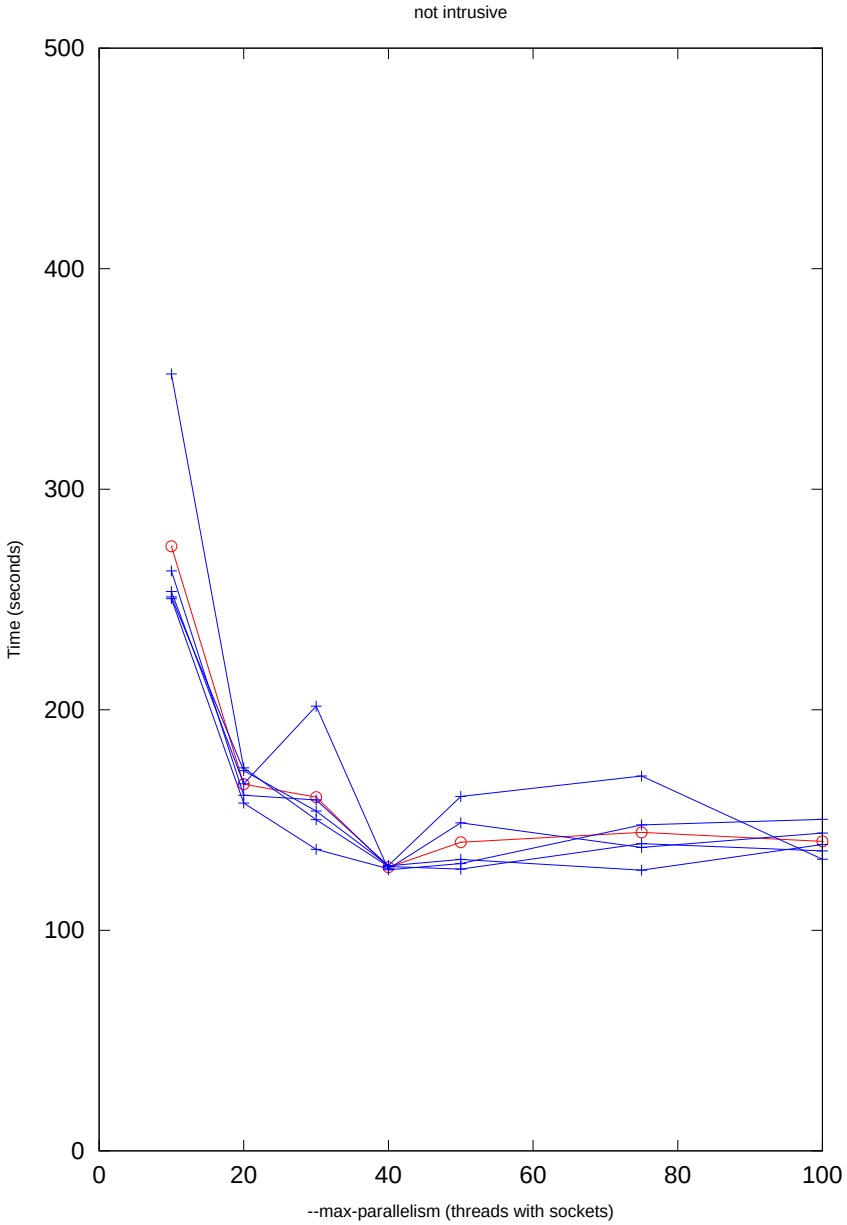
IV. Other Independent Results David Fifield graciously performed some independent scans on an older (slower) machine for comparison with the previous test results. We suspected there may be longer scan times and possibly different regressions for the timings given a slower machine:



–script "(safe or default) and not whois" For this particular test, we note that at –max-parallelism=10 we have almost a 100% increase in time compared to the earlier results. This appears to remain true for most data points. The scan time troughs at about 180 seconds. In the earlier results for this same script type, we hit a trough at about 100 seconds.

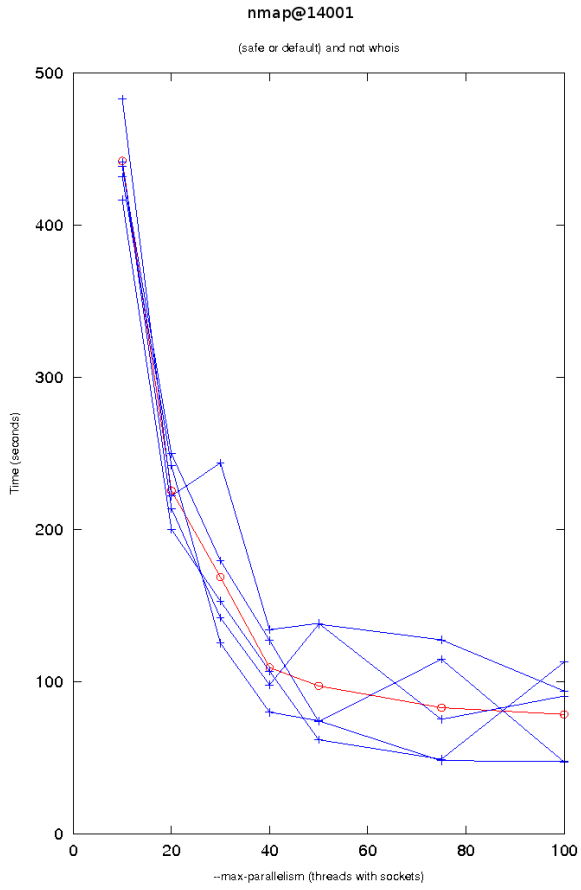


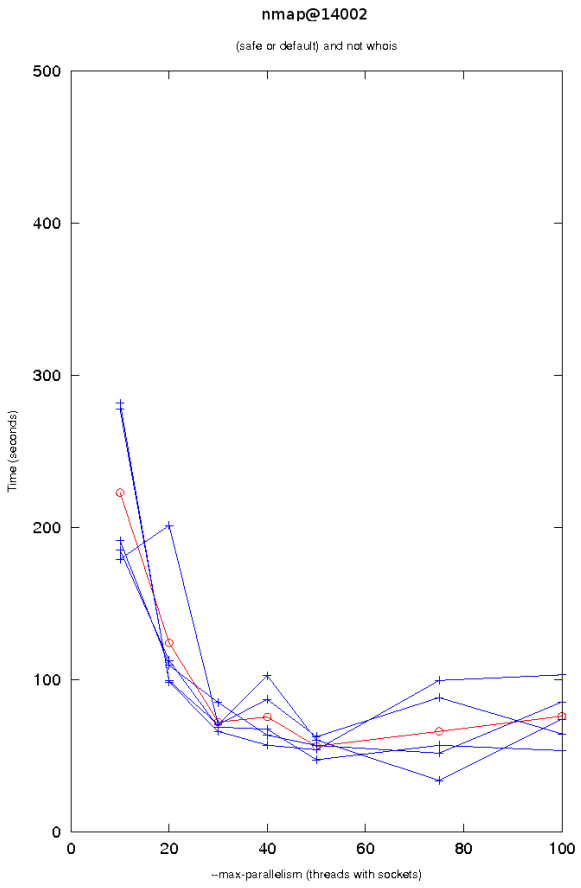
–**script "html-title"** Similar to the previous results, we receive very little benefit from increasing the parallelism, but likewise, we do not get penalized. Like the previous graph, this machine appears to have taken 100% more time to complete compared to the earlier results (40 seconds versus 80 seconds).



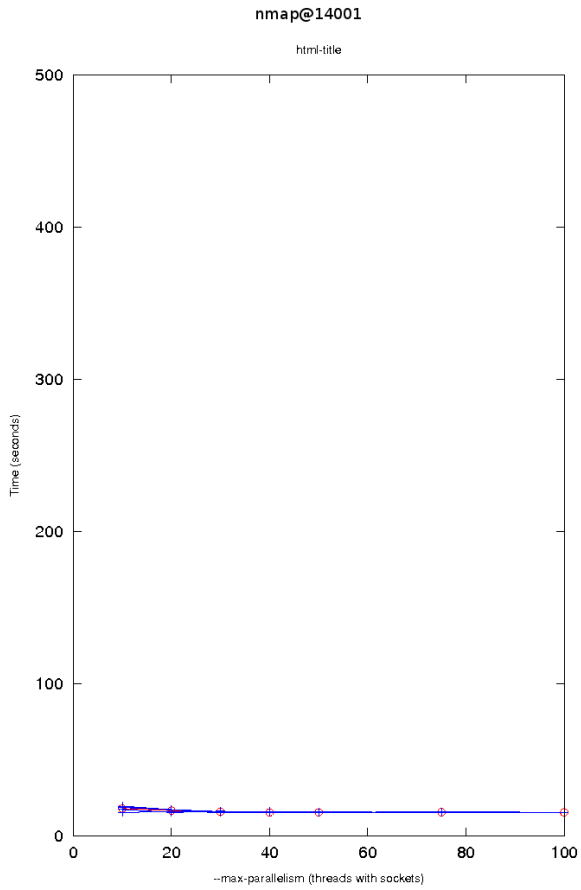
–**script "not intrusive"** Here we see that the graph has an exponential decay but more sharply decays to its expected value. It is interesting to note that at –max-parallelism=10, both this result and the previous have similar scan times. However, at higher parallelism, the expected value for this result is much higher (around 150 seconds versus 50 seconds).

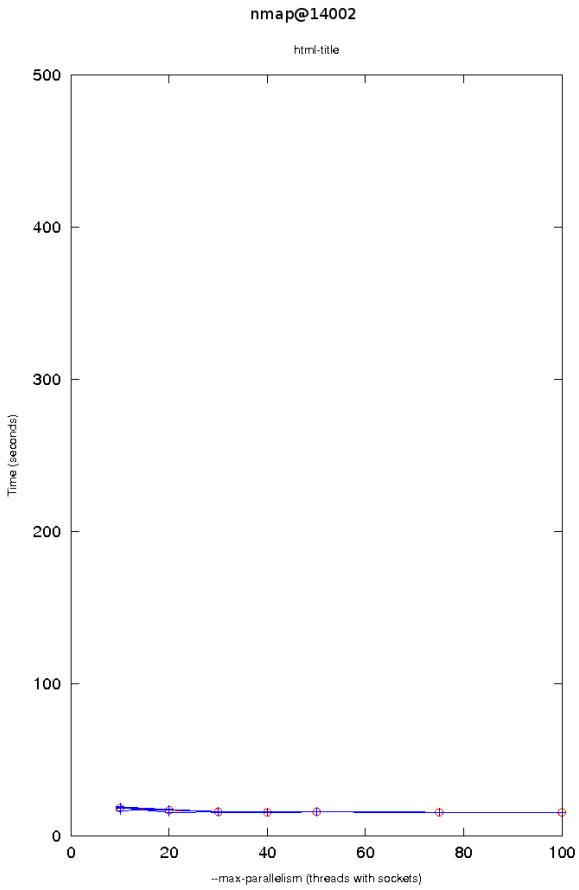
V. Revision 14002 This revision to Nmap is of particular interest for our testing. It corrected a problem where threads which finished for any reason, including errors, would hold onto open sockets until finally garbage collected. This caused increased wait times for scripts waiting to obtain an open socket. This revision closes all open sockets on dead scripts. What follows are graphs comparing the performance difference between the two revisions.



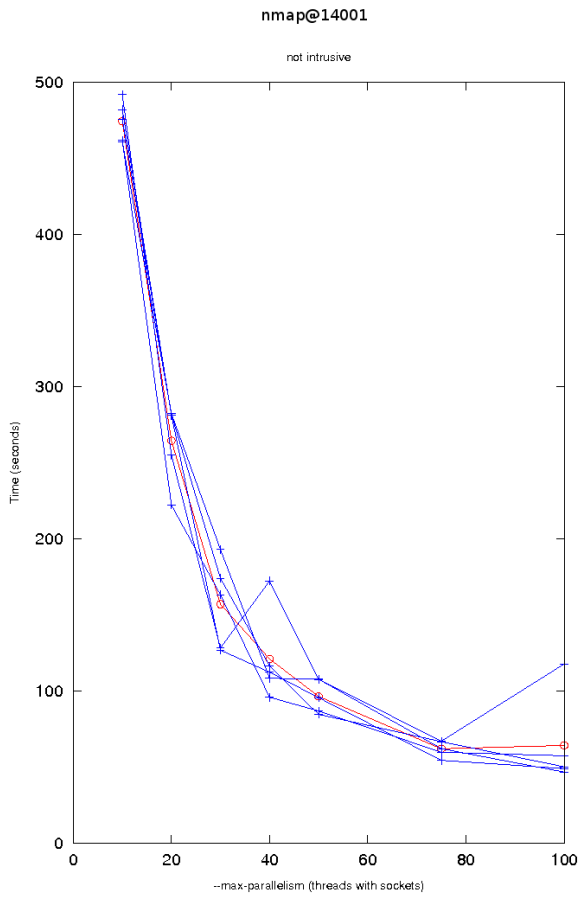


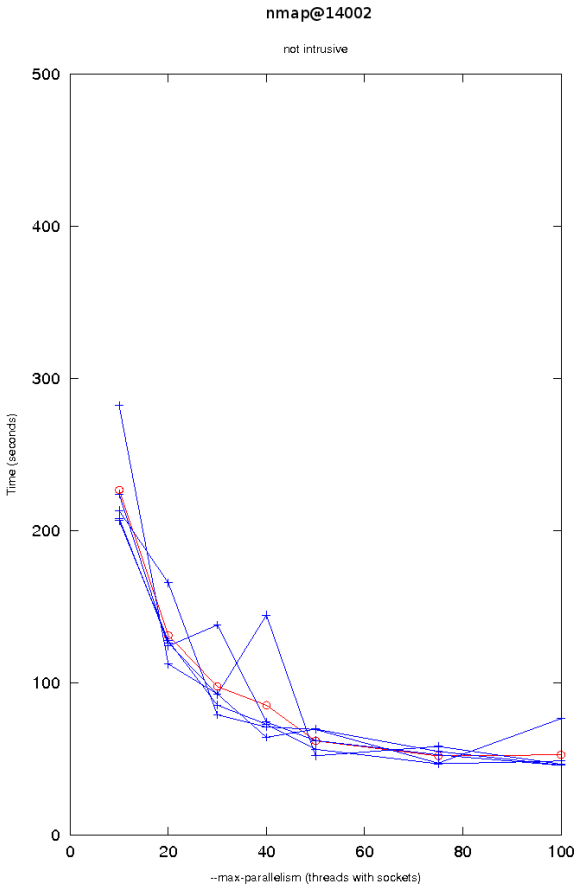
–**script "(safe or default) and not whois"** Here, you can see an approximate 50% decrease in scanning time between the two revisions.





–**script "html-title"** Again, html-title.nse appears to not be affected by this problem. Because of the reduced number of scripts (data), it is easier for dead scripts to be collected faster.





–**script "not intrusive"** This comparison shows a marked difference between the two revisions. Immediately releasing the resources of finished scripts allows scripts to obtain sockets quickly.

VI. Conclusions

We have seen from these tests that we get a good improvement in scan times at a parallelism of around 40. At that point we get diminished benefits from any further increases. The type of curve is generally consistent across machines; for smaller scans we see little or no benefit while for very large scans we get an exponential decrease in scan time.

I have committed a patch, revision 14264, that thus increases the default parallelism to a more ideal 40 instead of 10. One should expect the timings for NSE Scans to decrease for most scans. Of course, users can still manually increase or decrease the parallelism as their needs require.

In the future, it would be worthwhile to explore methods of dynamically locating an ideal parallelism. While scans typically did not gain huge benefits past a parallelism of 40, this predetermined ideal number could change in the future and a method of determining the ideal value dynamically would be more appropriate.

In conclusion, these tests showed a consistent improvement in scan times at a parallelism of 40 and we now set the default parallelism to this value. Users can expect Nmap's NSE scans to complete much faster now with this tuned default parallelism. Users should also feel encouraged to increase the parallelism if a scan seems to take too long.