Towards a GPU-Accelerated Domain Name System

(Abstract)

Matthäus Wander
Universität Duisburg-Essen
Duisburg, Germany
matthaeus.wander@uni-due.de

Johannes Brüderl
Universität Duisburg-Essen
Duisburg, Germany
johannes.bruederl@uni-due.de

1. INTRODUCTION

Authoritative servers in the *Domain Name System* (DNS) constitute the backbone of the Internet naming infrastructure. Scalability is one of the key issues to serve a large client population. The *Domain Name System Security Extensions* (DNSSEC) put additional burden on name servers due to larger message sizes and more complex server logic [1]. The NSEC3 secure denial of existence [6] is among the DNSSEC operations that can quickly exhaust the CPU and thus reduce the server capacity. Whenever a client queries for a non-existing domain name or resource record, a DNSSEC server with NSEC3 needs to compute the hash value of the query name to construct the proof of non-existence. The NSEC3 hash function is parametrized with the number of hash iterations, which serves as a trade-off between CPU load and the privacy assurance of NSEC3. With weak iterations, attackers can reverse the NSEC3 hash values and discover the contents of the DNS database via *zone enumeration*, which is what NSEC3 was supposed to prevent [2]. Prior work has shown that NSEC3 hash attacks can be carried out efficiently with GPUs [7].

We suggest to explore the use of GPUs on the server side to improve the scalability of the DNS. Offloading heavyweight operations to the GPU gives us the potential to free the CPU for other tasks, thus lifting existing capacity boundaries at lower costs than with another CPU-based host. In case of NSEC3, the additional processing power of the GPU allows us to raise the hash iterations and thus improve the protection against zone enumeration.

2. DESIGN RATIONALE

GPUs provide vast computing power by parallelizing work with a manycore processor architecture. Several technical limitations need to be considered when attempting to accelerate DNSSEC servers with GPUs:

1. GPU threads are executed in bundles called *wavefront* (AMD) or *warp* (Nvidia). Each GPU stream processor in a wavefront executes the same instruction in lockstep. If branch divergence occurs, the scheduler will stall some threads, thus degrading performance. Prior work suggests that this is one of the main reasons why the GPU speedup of asymmetric cryptographic algorithms is moderate at best [5]. The NSEC3 hash function is our first choice for GPU offloading, as it uses fast integer arithmetics without branch divergence.

2. GPUs have dedicated memory, which requires to move data between device boundaries before and after execution. Furthermore, while a GPU has very fast per-thread memory, large data structures like DNS zones will fit into global memory only. Threads compete for access to global memory, which can slow down GPU programs significantly.

3. DNSSEC servers should respond to individual queries as quickly as possible, but GPUs are best suited to batch computations. This leads to the optimization problem of compiling appropriately sized batches versus the latency penalty incurred to each response.

3. PRELIMINARY RESULTS

We have implemented a partial DNSSEC server prototype in C++/OpenCL that offloads the NSEC3 hashing to a GPU. Experiments with an AMD R9 390 and an Nvidia GTX 970 show that a GPU handles DNSSEC/NSEC3 load with any iteration count between 0 and 2500 while saturating a Gigabit Ethernet link. The response latency is about 13–15 ms for 0 to 150 iterations. In comparison when running the OpenCL code on an Intel i5-3570K CPU, the server throughput drops below Gigabit speed for configurations with more than 100 hash iterations. With 2500 iterations the CPU throughput caps at 11% of the GPU server performance.

4. RESEARCH QUESTIONS

Although the preliminary results show that GPU offloading is a promising approach to improve the scalability of DNSSEC servers, a few open questions remain.

1. What is the best strategy for GPU offloading, e.g. opportunistic offloading above a certain load threshold?
2. What is the sweet spot of GPU batch sizes subject to hash iterations, query load and latency?
3. What benefit can be expected in a real-world scenario, i.e. with a full-featured DNSSEC server and mixed traffic consisting of regular and NSEC3 responses?
4. How to integrate GPU accelerators with existing server implementations?
5. Is it worthwhile to offload online signing, other cryptographic methods like NSEC5 [4] or DNS zone lookups to the GPU?
6. Can we lower the per-response latency by using parallelizable hash functions like Keccak [3] instead of sequantially iterated hashing?
7. Are CPUs with *Integrated Graphics Processors* (IGPs) favorable over dedicated GPUs due to zero-copy memory usage?

5. REFERENCES

[1] B. Ager, H. Dreger, and A. Feldmann. Predicting the DNSSEC overhead using DNS traces. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 1484–1489. IEEE, 2006.

[2] D. J. Bernstein. Breaking DNSSEC, Aug. 2009. Keynote lecture at Workshop on Offensive Technologies (WOOT).

[3] P.-L. Cayrel, G. Hoffmann, and M. Schneider. GPU Implementation of the Keccak Hash Function Family. In *International Conference on Information Security and Assurance*, pages 33–42. Springer, 2011.

[4] S. Goldberg, M. Naor, D. Papadopoulos, L. Reyzin, S. Vasant, and A. Ziv. NSEC5: Provably Preventing DNSSEC Zone Enumeration. *IACR Cryptology ePrint Archive*, 2014:582, 2014.

[5] O. Harrison and J. Waldron. Efficient Acceleration of Asymmetric Cryptography on Graphics Hardware. In *International Conference on Cryptology in Africa*, pages 350–367. Springer, 2009.

[6] B. Laurie, G. Sisson, R. Arends, and D. Blacka. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155, Mar. 2008.

[7] M. Wander, L. Schwittmann, C. Boelmann, and T. Weis. GPU-Based NSEC3 Hash Breaking. In *Network Computing and Applications (NCA), 2014 IEEE 13th International Symposium on*, pages 137–144. IEEE, 2014.