ABSTRACT

The SHA-1 hashing algorithm is widely used for identification and authentication purposes. In digital evidence forensics, hashing is used for verification of integrity and correctness of evidence. But there is a huge computational complexity and delay due to its data dependency in the critical path iterations. To deal with the problem a new algorithm RSHA-1 is proposed which serves the same purpose as SHA-1 with less computational complexity and delay by using the power of GPU's. The target is not only to reduce the delay but also to lead green computing by saving power consumption.

The algorithm implements recursive hash to break the chain dependencies of the standard hash function. We discuss here the theoretical foundation for the work and the performance implications. The algorithm is implemented using the OpenMP API as well as CUDA on GPU's.

MOTIVATION

Traditional Cryptographic hashing algorithms like MD5, SHA-1, SHA-2 are mainly serially implemented and are used in various applications like Digital Signatures, Forensics, SSL protocol, authentication etc. The computation of hash value requires lot of execution time.

In forensics, the hash process is normally used during acquisition of the evidence, during verification of the forensic image, and again at the end of the examination to ensure the integrity of the data and forensic processing. These algorithms are also currently used to validate the integrity of downloaded files in information technology applications.

The amount of data often exceeds 1 terabyte and it takes too much time to calculate the hash code for such heavy files.

So the power of multiprocessors on a GPU machine can be exploited to parallelize the hashing algorithms which can lead to its fast and secure implementation.

EXPERIMENTAL RESULTS

In order to assess the new framework properly, the algorithm was implemented in both using OpenMP API and CUDA on GCC infrastructure. Firstly the experiment was conducted in OpenMP and following readings were found:

- Speedup / (no.of processors)
- Efficiency = Speedup / (no.of processors)

In OpenMP, it was observed that RSHA-1 can give good results for producing hash code for the heavy files. So, we compiled the serial implementation of SHA-1 on CPU and GPU and first and found that with nvcc, code gives a speedup of around 5x when compared with sequential SHA-1 using OpenMP API. The following table shows the results of same code on different file and different core machine, so as to analyze the speedup gained.

<table>
<thead>
<tr>
<th>File Size</th>
<th>Execution Time (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32MB</td>
<td>2.1252</td>
</tr>
<tr>
<td>64MB</td>
<td>4.2102</td>
</tr>
<tr>
<td>128MB</td>
<td>8.3570</td>
</tr>
<tr>
<td>256MB</td>
<td>14.1567</td>
</tr>
<tr>
<td>512MB</td>
<td>28.6915</td>
</tr>
<tr>
<td>1GB</td>
<td>57.2625</td>
</tr>
</tbody>
</table>

Algorithm 1: RSHA-1 Algorithm

1. Append padding bits to the original message if required.
2. Calculate the length of the original message and append it to the end of the message.
3. Split new message space M into blocks M[i] of 512 bits each.
4. Cut-out the dependence among hash values H using bit (i) for all blocks M[i].
5. Calculate H(M[i]) of 160 bits for each M[i] to generate H[i].
6. Combine H[i], ..., H[i], to form a single message.
7. Use recursive hash as similar to H(M[i]) to produce an output hash value.

Complexity Calculation

Let the Complexity of calculating $H(M)$ for hash value of each message blocks is $Cmplx(H(M))$, and the number of original message blocks as $P$, with $n$ (512)- bit, $m$ (160)- bit length of hash value and $k$=log16/5p.

So the computational complexity for RSHA-1 can be given by $f(R)$:

$$f(R) = (p + n_c) \cdot Cmplx(H(M))$$

where $n_c$ is number of calls to $H(M)$ due to reduction and is given by:

$$n_c = \left\lfloor \frac{m}{n} \right\rfloor + \left\lfloor \frac{m}{n} \right\rfloor - 1$$

Although RSHA-1 has more computations than that of standard SHA-1 but RSHA-1 is more secure than standard SHA-1 due its recursive nature.

Performance Measures

Let the time taken by standard SHA-1 to calculate hash value of each message block is $T(M)$

The processing time of standard SHA-1 is given by:

$$T = \frac{p}{T(M)}$$

and the processing time of RSHA-1 is given by:

$$T = \left\lfloor \log_{p+1} P \right\rfloor T(M)$$

So, Speedup = $T / T$, $T = \left\lfloor (p/\log(p+1)) \right\rfloor T(M)$

and Efficiency = Speedup / (no.of processors)

Where $p$ is number of original message blocks with $n$ (512)- bit, $m$ (160) is length of hash value and $k$=log16/5p.

Hence, as the size of data and the number of processors increases, the performance shown by the algorithm is better than the standard SHA-1.