Hashcash Parallelization on GPGPU using OpenCL

Etienne Servais

MINES ParisTech, Centre de Recherche en Informatique

April 19, 2011

Table of Contents

1. Hashcash overview
   - The hashcash protocol
   - Stamp format in current version
   - Current CPU implementation performances
2. OpenCL
3. My implementation of Hashcash
   - source code survey
4. My results
   - Performances
   - Economical analyze
   - Stamp record!

Asymetric *proof-of-work* algorithm based on burning CPU cycles.

Goals :

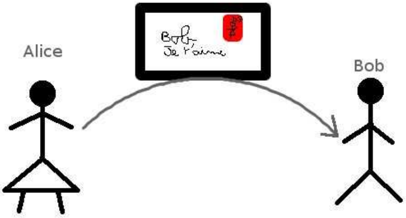
- Spam-killer :

  E-mail header :

  ```
  From:  Calvin <calvin@comics.net>
  To:  Hobbes <hobbes@comics.net>
  Subject:  Suzy Derkins
  Date:  19 Jan 2038 11:59:59 +0000
  X-Hashcash:  1:24:380119:hobbes@comics.net::000000000FE
  ```

- DoS defense.

The hashcash protocol



Stamp example

```
1:24:110309:bob@comics.net::M/42qrTP4ANgmSSs:003oMpI
```

Stamp format in current version

```
ver:bits:date:resource:rand:counter
```

Where

- ver = 1
- bits = Number of bits of the partial-preimage
- date = YYMMDD
- resource = IP, email adress. . .
- rand = random string, avoids getting twice the same stamp.
- counter = string used to find the preimage.

Current CPU implementation performances

Architecture	Collision tests per sec.	estimate for a 20 bits stamp	estimate for a 30 bits stamp
AMD Turion(tm) X2 Ultra Dual-Core Mobile ZM-82 at 2.2GHz	4508800	255 ms	235 s
Intel Xeon at 2.00GHz	4131200	253 ms	260 s
Intel Atom N270 at 1.60GHz	2224000	603 ms	460 s
Intel Pentium M at 1.70GHz	3478400	305 ms	310 s
AMD Phenom(tm) II X4 955 Processor at 3.20GHz	6144000	171 ms	175 s

Figure: Hashcash calculation time on different computers.

Cross platform API to C.

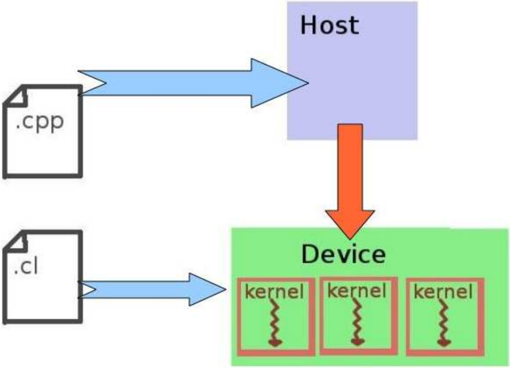
KHRONOS GROUP

Specifications written by the Khronos Group regrouping :

-
- AMD
- IBM
- intel
- NVIDIA
- . . .

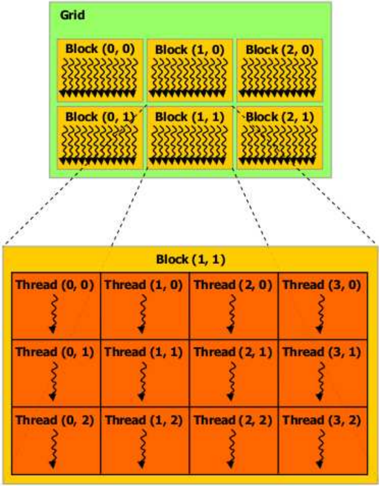
Specifications first published December 8, 2008.

2 parts in an OpenCL program :



3 levels of parallelism :

- Work-items (an instance of a kernel, a *thread* in CUDA) contained inside
- Work-groups (*thread-blocks* in CUDA) sharing data contained inside
- NDRanges which is a 1-to-3 dimensional container.



Host code :

SHA-1 stamp construction :

$\boxed{\texttt{1:24:380119:hobbes@comics.net::0}\boxed{\texttt{00000000}}}\texttt{FE4EA5E9}$

Device code :

Host code :

Device code :

SHA-1 stamp construction :

```
1:24:380119:hobbes@comics.net::000000000 FE4E A5E9
```

Based on FIPS PUB 180-1

```
for(i = 20; i < 40; i++)
{
   W[i & 0x0f] = rotateLeft(W[(i-3) & 0x0f]
     ^ W[(i-8) & 0x0f] ^ W[(i-14) & 0x0f]
     ^ W[(i-16) & 0x0f], 1);
   temp = rotateLeft(A, 5) +
     (B ^ C ^ D) + E + W[i & 0x0f] + K1;
   E = D;
   D = C;
   C = rotateLeft(B, 30);
   B = A;
   A = temp;
}
```

Full loop unrolling

```
//i=25
W[9] = rotateLeft(W[6] ^ W[1] ^ W[11] ^ W[9], 1);
temp = rotateLeft(A, 5) +
   (B ^ C ^ D) + E + W[9] + K1;
E = D;
D = C;
C = rotateLeft(B, 30);
B = A;
A = temp;
```

Array scalarization

```
//i=25
W9 = rotateLeft(W6 ^ W1 ^ Wb ^ W9, 1);
temp = rotateLeft(A, 5) +
  (B ^ C ^ D) + E + W9 + K1;
E = D;
D = C;
C = rotateLeft(B, 30);
B = A;
A = temp;
```

Copy propagation

```
//i=25
W9 = rotateLeft(W6 ^ W1 ^ Wb ^ W9, 1);
t1 = rotateLeft(t0, 5) +
  (t2 ^ C1 ^ C0) + C2 + W9 + K1;
C2 = rotateLeft(t2, 30);
```

Standard implementations has about 3000 basic operations per SHA-1 :

- 656 assignations

In my version, only 1200 per SHA-1 !

- 228 assignations

Standard implementations has about 3000 basic operations per SHA-1 :

- 224 rotations

In my version, only 1200 per SHA-1 !

- 220 rotations

Standard implementations has about 3000 basic operations per SHA-1 :

- 1886 logical and aritmetical operations

In my version, only 1200 per SHA-1 !

- 752 logical and arithmetical operations

Standard implementations has about 3000 basic operations per SHA-1 :

- 432 array accesses

In my version, only 1200 per SHA-1 !

Operations	Initially	In my implementation
Assignations	656	228
Rotations	224	220
Logical and arithmetical operations	1886	752
Array Accesses	432	0
Total	3198	1200

Figure: Comparison of official and optimised version of SHA-1.

Architecture	Operations per sec.	SHA-1 per sec.	Prix en euros
NVIDIA Tesla C2050 14 GPU at 1147 MHz (448 CUDA cores)	513 G $2^{38.90}$	424 M $2^{28.66}$	2350
NVIDIA GeForce 8800 GTX 16 GPU at 1350 MHz (128 CUDA cores)	173 G $2^{37.33}$	142 M $2^{27.08}$	40 (ebay.fr)
AMD Phenom(tm) II X4 955 Processor at 3.20 GHz	13 G $2^{33.58}$	(20 M) $2^{24.26}$	150

Figure: SHA-1 performances on different devices.

- 1 response per 12,500,000 emails (november 2008) ;

You earn per year about :

$$P = 365,25 \times 24 \times 3600 \times 2^{27.08-b} \times \frac{10}{12500000} - 40 - 365,25 \times 24 \times 0,280 \times 0,12$$

b : number of required bits.

- Selling products 10€ each ;

You earn per year about :

$$P = 365,25 \times 24 \times 3600 \times 2^{27.08-b} \times \frac{10}{12500000} - 40 - 365,25 \times 24 \times 0,280 \times 0,12$$

b : number of required bits.

- Buying a GeForce 8800 GTX (40€) ;

You earn per year about :

$$P = 365,25 \times 24 \times 3600 \times 2^{27.08-b} \times \frac{10}{12500000} - 40 - 365,25 \times 24 \times 0,280 \times 0,12$$

b : number of required bits.

- Computer power is about 280W (dev configuration) ;

You earn per year about :

$$P = 365,25 \times 24 \times 3600 \times 2^{27.08-b} \times \frac{10}{12500000} - 40 - 365,25 \times 24 \times 0,280 \times 0,12$$

b : number of required bits.

- In France, 1 kWh costs about 0,12€ ;

You earn per year about :

$$P = 365,25 \times 24 \times 3600 \times 2^{27.08-b} \times \frac{10}{12500000} - 40 - 365,25 \times 24 \times 0,280 \times 0,12$$

b : number of required bits.

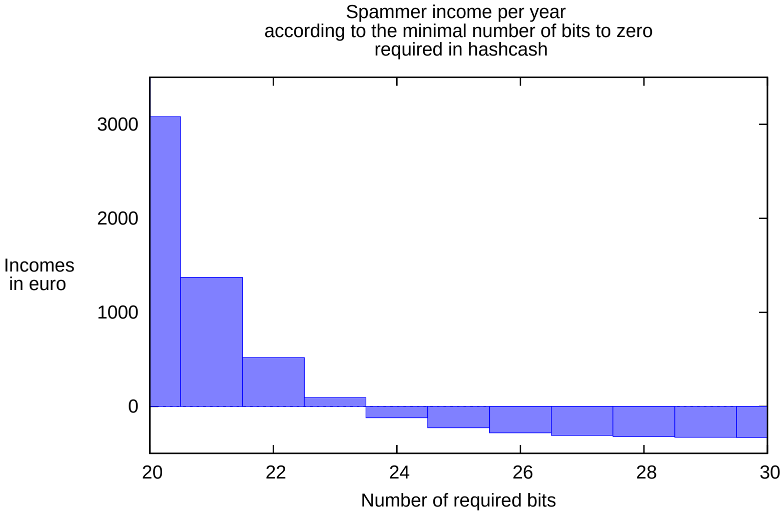
- 1 response per 12,500,000 emails (november 2008) ;
- Selling products 10€ each ;
- Buying a GeForce 8800 GTX (40€) ;
- Computer power is about 280W (dev configuration) ;
- In France, 1 kWh costs about 0,12€ ;

You earn per year about :

$$P = 365,25 \times 24 \times 3600 \times 2^{27.08-b} \times \frac{10}{12500000} - 40 - 365,25 \times 24 \times 0,280 \times 0,12$$

b : number of required bits.

Economical analyze



Stamp record!

A 48 bits stamp!

```
1:48:110416:etienne@cri.fr:::000A2F00000063BF012
```

Obtained in 1266748 seconds (about 14 days, 10 hours)

Stamp record!

Thank you for your attention!