ECM

Adéla Haníková

12. ledna 2015

Obsah

1 Úvod **1**

2 ECM **1**
2.1 Edwardsovy křivky 2
2.2 Algoritmus 2

3 Aritmetika **3**
3.1 Montgomeryho reprezentace 3

4 OpenCL **3**

5 Program **4**
5.1 Soubory 4
5.2 Výpočet 4
5.3 Parametry 5

1 Úvod

ECM neboli metoda eliptických křivek je v současnosti asymptoticky nejrychlejší faktorizační algoritmus, jehož složitost závisí na velikosti hledaného dělitele. Je to také nejlepší algoritmus pro faktorizace, které jsou potřeba pro běh číselného síta, i z toho důvodu je snaha tento algoritmus stále zdokonalovat.

V této implementaci jsou využity Edwardsovy křivky, na nichž můžeme sčítat body rychleji. Jednou z výhod ECM proti Pollardově p-1 metodě je také to, že se číslo můžeme pokusit faktorizovat na více různých křivkách, jejichž grupy budou mít různé (náhodné) řády. To předurčuje tuto metodu k paralelizaci výpočtů, která je zde implementována pomocí standardu OpenCL.

V následujícím textu N značí číslo, které chceme faktorizovat.

2 ECM

Algoritmus pro faktorizaci pomocí eliptických křivek byl poprvé zmíněn H. W. Lenstrou v článku [3] roku 1987. Jedná se v podstatě o Pollardovu (p-1)-metodu, ve které je multiplikativní grupa nahrazena aditivní grupou bodů na eliptické

křivce. Výhoda této metody spočívá v tom, že pro faktorizaci jednoho N, můžeme využít více křivek s různými řády a máme tudíž větší pravděpodobnost nalézt dělitele.

Roku 2007 představil Edwards v článku [1] křivky, nyní nazývané Edwardsovy, které jsou s eliptickými v jistém smyslu ekvivalentní a zároveň umožňují implementovat sčítání bodů rychleji. Současné nejrychlejší implementace tedy využívají právě Edwardsovy křivky, viz [2].

2.1 Edwardsovy křivky

Rovnice Edwardsovy křivky nad tělesem $\mathbb{F}$ charakteristiky různé od dvou je

$$x^2 + y^2 = 1 + dx^2y^2, \quad d \in \mathbb{F} \setminus \{0, 1\}.$$

V článcích se objevuje 5 způsobů, jak z předpisu křivky získat body, výsledkem je afinní, projektivní, invertovaná, rozšířená nebo kompletní množina bodů. Pro každou z nich platí jiné vzorečky a tedy jiné výhody, nevýhody a počet potřebných násobení. Stručně zpracovaný přehled naleznete v článku [2].

Nejpřímější metoda je vytvořit množinu afinních bodů:

$$\{(x, y) \in \mathbb{F} \times \mathbb{F} : x^2 + y^2 = 1 + dx^2y^2\}.$$

V takovém případě je bod $(0, 1)$ neutrální prvek vzhledem ke sčítání a opačný prvek k bodu (x, y) je $(-x, y)$. Vzoreček pro sečtení dvou různých bodů i zdvojnásobení bodu je stejný a pokud d není čtverec v $\mathbb{F}$, je tento vzoreček úplný (definovaný v každém bodě).

Výhodnější pro implementaci je ale rozšířená množina bodů:

$$\{(X : Y : Z : T) \in \mathbb{P}^3 : X^2 + Y^2 = Z^2 + dT^2 \text{ a } XY = ZT\}.$$

Afinní body se zobrazí na body $\mathbb{P}^3$ následujícím předpisem: $(x, y) \to (x : y : 1 : xy)$ a přibudou čtyři body v nekonečnu, pokud d je čtverec: $(0 : \pm\sqrt{d} : 0 : 1)$ a $(1 : 0 : 0 : \pm\sqrt{1/d})$. V článku [5] pak byly odvozeny vzorečky pro sečtení bodů $(X1 : Y1 : Z1 : T1)$ a $(X2 : Y2 : Z2 : T2)$, které vyžadují 9 násobení:

$$A = X1 * X2,\ B = Y1 * Y2,\ C = Z1 * T2,\ D = T1 * Z2$$
$$E = D + C,\ F = (X1 - Y1) * (X2 + Y2) + B - A,\ G = B + A,\ H = D - C$$
$$X3 = E * F,\ Y3 = G * H,\ T3 = E * H,\ Z3 = F * G.$$

2.2 Algoritmus

Samotný algoritmus je v principu velmi jednoduchý. Předpokládejme, že $N = p * q$ a vygenerujme křivku E a bod P nad $\mathbb{Z}_p$. Budeme sčítat bod P sám se sebou a doufat, že narazíme na neutrální prvek, tedy na bod $(0, 1)$. Problém je, že p je námi hledané prvočíslo, které neznáme. Budeme tedy ve skutečnosti počítat modulo N, v jakési nadbytečné reprezentaci čísel modulo p. Víme, že pokud výsledkem součtu je neutrální prvek, jeho x-ová souřadnice je rovna 0 modulo p a tedy $p|NSD(N, x)$.

Rozklad čísla N na jedné křivce pak probíhá následovně:

1. Vygenerujeme náhodnou křivku nad $\mathbb{Z}/N\mathbb{Z}$ tvaru $x^2 + y^2 = 1 + dx^2y^2$ a netriviální bod P ležící na této křivce.

2. Spočteme $Q = [e]P = \underbrace{P \oplus \cdots \oplus P}_{e\text{-krát}}$, kde e je součin malých prvočísel. Tento součet počítáme postupně ve for-cyklu, tak, že v i-té iteraci spočteme $P := [p_i]P$.

3. Spočteme $NSD(Q_x, N)$, kde Q_x je x-ová souřadnice výsledného bodu Q.

3 Aritmetika

Vzhledem k tomu, že celý faktorizační algoritmus probíhá na grafické kartě a je psán v jazyce OpenCL C, bylo potřeba implementovat i základní aritmetiku velkých čísel. Velké číslo je uloženo v poli délky 8 jehož prvky jsou `unsigned integer` a není brán ohled na znaménko - číslo se vždy reprezentuje jako kladné. Tato struktura je v hostitelském kódu pojmenována `my_mpz` a v kódu kernelu `mpz`. Pro základní operace jsou, vzhledem k velikosti čísel, použity školské algoritmy.

3.1 Montgomeryho reprezentace

Časově náročné je ovšem počítání souřadnic bodů modulo N. Výrazného zrychlení se dosáhne použitím Montgomeryho reprezentace [4]. Stručně řečeno, je každé $x \mod N$ je reprezentováno hodnotou $x' = Rx \mod N$, kde $R > N$ je námi pevně zvolené číslo nesoudělné s N. Protože jsou R a N nesoudělná, existují čísla R^{-1} a N': $RR^{-1} - NN' = 1$

Algoritmus sčítání se nijak nemění, sečtu-li dva reprezentanty, dostanu reprezentanta součtu. Pro vynásobení dvou reprezentantů x, y modulo N při využití Montgomeryho reprezentace se použijí následující vzorce:

$q = (xy \mod R)N' \mod R;$
$a = (x + qN)/R;$
if $a > N$ **then** $a = a - n;$
return a

Musíme sice tedy využít vícekrát algoritmus pro vynásobení dvou čísel, na druhou stranu potřebujeme dělit a modulit pouze číslem R. To je typicky voleno jako dostatečně velká mocnina 2 a dělení i modulo se provedou bitovými posuny. Srozumitelný popis lze najít v článku [6].

4 OpenCL

OpenCL je otevřený standard pro paralelní programování heterogenních počítačových systémů a obsahuje tři hlavní části: abstraktní modely určující požadované vlastnosti a chování zařízení OpenCL, OpenCL framework a specifikaci programovacího jazyka OpenCL C. Pro svoji abstraktnost a kompabilitu se začíná těšit stále větší oblibě a proto byl vybrán k implementaci ECM.

Programování v OpenCL je rozděleno na dvě části, z nichž jedna je spouštěna v rámci hostitelského systému a druhá na OpenCL zařízení, kterým hostitelský systém disponuje, typicky na grafické kartě. Zdrojový kód pro OpenCL zařízení je psán v jazyce OpenCL C, který je postaven na normě C99 jazyka C. Oproti

C99 obsahuje některá rozšíření, jako například kvalifikátory adresního prostoru, vektorové datové typy nebo kernelové funkce, ale také některá omezení jako například zákaz rekurzivního volání funkcí či nedostupnost valné většiny hlavičkových souborů standardní knihovny jazyka C.

5 Program

Pro vytvoření programu využijte přiložený makefile. Kromě standardních knihoven (`iostream, string, fstream, cmath`), jsou potřeba také knihovny GMP a OpenCL.

5.1 Soubory

Deklarace všech funkcí třídy ECM i s jejich popisem jsou v souboru ECM.h, jejich definice jsou ovšem rozděleny do více souborů.

`ECM.cpp` obsahuje definice funkcí, které tvoří hlavní kostru algoritmu.

`ECM_setup.cpp` obsahuje funkce pro předání parametrů ECM a funkci, která zajišťuje nastavení OpenCL.

`ECM_helpers.cpp` obsahuje pomocné funkce, jako například převod do a z Montgomeryho tvaru, převod na typ `my_mpz` nebo generování prvočísel.

`ECM_kernels.cl` je soubor se zdrojovým kódem kernelu, obsahuje celý algoritmus ECM.

`ECM_parameters.cpp` obsahuje samostatnou třídu `ECM_parameters`, která se stará o načtení parametrů z příkazové řádky, jsou-li zadány.

5.2 Výpočet

Program je rozdělen do dvou částí; první, přípravná a závěrečná, část probíhá na procesoru, zatímco druhá, výpočetní část na OpenCL zařízení.

Při spuštění programu se nejprve pomocí třídy `ECMparameters` načtou zadané parametry, pokud byly zadány, v opačném případně je o ně zažádáno po spuštění programu. Následně se předají třídě `ECM`. Ta nejprve provede vše nezbytné pro OpenCL - najde platformy a příslušná zařízení, vytvoří kontext, příkazovou frontu a přeloží kernel pro vybrané zařízení. Pokud je na dané platformě více dostupných OpenCL zařízení, musí zvolit uživatel, které se má použít.

Následně jsou provedeny předvýpočty pro ECM - vygenerují se všechna prvočísla do dané meze B1, najde se parametr R pro Montgomeryho reprezentaci čísel, vygenerují se parametry křivek a body a převedou se do Montgomeryho reprezentace. Spočítá se a převede také N' pro Montgomeryho násobení. Všechny tyto hodnoty jsou pak načteny do OpenCL bufferů, které jsou zkopírovány na OpenCL zařízení a je spuštěn kernel.

Na OpenCL zařízení se pak vykoná kód kernelu, tedy pouze samotné sčítání bodu na křivce. V každém vlákně se provádí totožný výpočet, jen na jiné křivce. Výstup kernelu je pouze x-ová souřadnice výsledného bodu z každého vlákna.

Ta je předána zpátky třídě ECM, která ji převede zpět na typ mpz a pomocí knihovny GMP spočte největší společný dělitel se zadaným číslem N. Pokud je výsledek netriviální, vypíše nalezeného dělitele.

5.3 Parametry

Program lze spustit bez parametrů, zeptá se poté na potřebné údaje po spuštění, nebo lze údaje zadat pomocí parametrů. I při spuštění programu s parametry je ale potřeba, aby uživatel zvolil, jaké z dostupných OpenCL zařízení se má ke spuštění použít, pokud jich je na dané platformě více.

Parametry programu jsou:

-n *číslo_pro_faktorizaci* [**-b1** *mez_b1*] [**-c** *počet_křivek*]

Program se pokusí najít dělitele čísla n.

-n *číslo_pro_faktorizaci* číslo pro faktorizaci zadané bez mezer

-b1 *mez_b1* mez pro první fázi algoritmu; implicitně 10000

-c *počet_křivek* počet křivek, které se mají vygenerovat a použít pro algoritmus; implicitně 100

Parametry pro faktorizaci více čísel:

-muln *počet_čísel čísla_pro_faktorizaci* [**-b1** *mez_b1*] [**-c** *počet_křivek*]

Program se pokusí postupně najít dělitele všech zadaných čísel. Mez B1 i počet křivek je stejný pro všechna čísla.

-muln *počet_čísel čísla_pro_faktorizaci* zadejte počet čísel pro faktorizaci a postupně všechna čísla oddělená mezerami

Vzorové vstupy jsou v souborech vstupX.txt, $X \in \{1 \dots 4\}$.

Reference

[1] Harold M. Edwards, *A normal form for elliptic curves*, Bulletin of the American Mathematical Society **44** (2007), 393-422. URL: `http://www.ams.org/bull/2007-44-03/S0273-0979-07-01153-6/home.html`

[2] Daniel J. Bernstein, Peter Birkner, Tanja Lange, Christiane Peters *ECM using Edwards curves* (2008). URL: `http://eprint.iacr.org/2008/016`

[3] Hendrik W. Lenstra, Jr., *Factoring integers with elliptic curves*, Annals of Mathematics **126** (1987), 649-673, ISSN 0003-486X. MR 89g:11125. URL: `https://openaccess.leidenuniv.nl/bitstream/1887/3826/1/346_086.pdf`

[4] Peter L. Montgomery, *Modular multiplication without trial division*, Mathematics of Computation **44** (1985), 519-521. URL: `http://www.ams.org/journals/mcom/1985-44-170/S0025-5718-1985-0777282-X/home.html`.

[5] Huseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, Ed Dawson, *Twisted Edwards curves revisited*, in Asiacrypt 2008 (2008). URL: `http://eprint.iacr.org/2008/522`.

[6] Professor Dr. D. J. Guan *Montgomery Algorithm for Modular Multiplication* (2003) URL: `http://guan.cse.nsysu.edu.tw/note/montg.pdf`.