POST-SIEVING ON GPUS

Andrea Miele[1],
Joppe W. Bos[2],
Thorsten Kleinjung[1],
Arjen K. Lenstra[1]

[1]LACAL, EPFL, Lausanne, Switzerland
[2]NXP Semiconductors, Leuven, Belgium

GPU COMPUTING HISTORY

- 90s: parallel devices for graphics pipeline, transforming 3D scene description (triangles) into 2D pixel frame to put on screen

- Many cores, each specialized for particular pipeline stage. Became more and more programmable across the years…

- **2006: full programmability plus general purpose programming models (CUDA, OpenCL): GPGPU**

OUR STUDY

- We investigated the use of NVIDIA GPUs supporting CUDA as accelerators for the number field sieve
- We targeted FERMI GPUs (5 years old now…)
- Applies to massive parallel PKC apps based on modular arith
- **GPU integer performance has been dropping dramatically, but I talked with NVIDIA people and there may be hope…**

NUMBER FIELD SIEVE (NFS)

- NFS: asymptotically fastest known factoring algorithm
- RSA 768-bit modulus factored with NFS (2010)
- Idea: to factor an odd composite **n**, find integer solutions **x**, **y** to $x^2 \equiv y^2 \bmod n$ such that $x \neq \pm y \bmod n$
- Two main steps:

  Relation collection: find smooth integers, ≈90% of total time

  Linear algebra step: find solutions (x,y), ≈10% of total time

NFS RELATIONS

- Two positive integer smoothness bounds: $\mathbf{B_{ra}}$, $\mathbf{B_{al}}$
- Irreducible $\mathbf{f_{ra}(X)}$, $\mathbf{f_{al}(X)}$ of degree **l** and **d** small (**d=6**)
- **Relation**: **(a,b)** with **a**,**b** co-prime integers (**b>0**) such that

1. $\mathbf{b \cdot f_{ra}(a/b)}$ is $\mathbf{B_{ra}}$-smooth except at most **3** primes in $(\mathbf{B_{ra}}, \mathbf{B_L}]$
2. $\mathbf{b^d \cdot f_{al}(a/b)}$ is $\mathbf{B_{al}}$-smooth except at most **4** primes in $(\mathbf{B_{al}}, \mathbf{B_L}]$

COLLECT RELATIONS

- SIEVING find pairs **(a,b)** such that:

  $b \cdot f_{ra}(a/b) = cg$ (rational side)

  $b^d \cdot f_{al}(a/b) = eh$ (algebraic side)

  where **c** is B_{ra}-smooth, **e** is B_{al}-smooth and $g \leq B_L^3$, $h \leq B_L^4$ ("cofactors")

- **POST-SIEVING:**

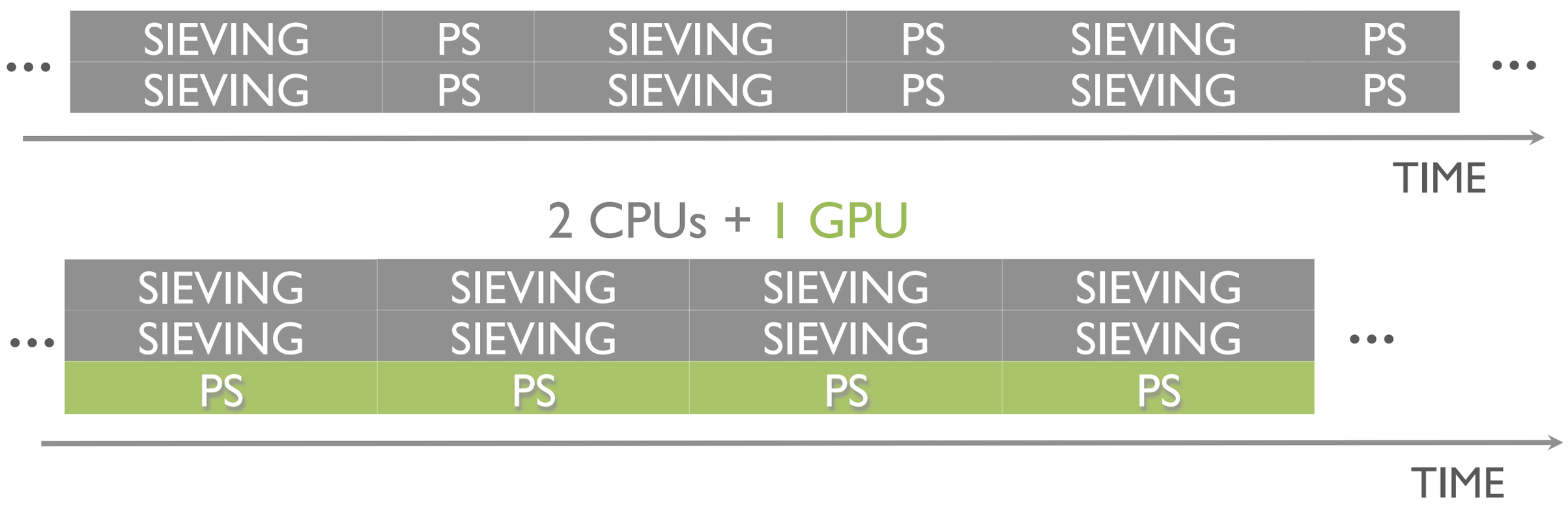
  1 Compute $b \cdot f_{ra}(a/b)$ and $b^d \cdot f_{al}(a/b)$

  2 Factor $b \cdot f_{ra}(a/b)$ and $b^d \cdot f_{al}(a/b)$ to check if prime divisors are at most B_L

  EMBARRASSINGLY PARALLEL!

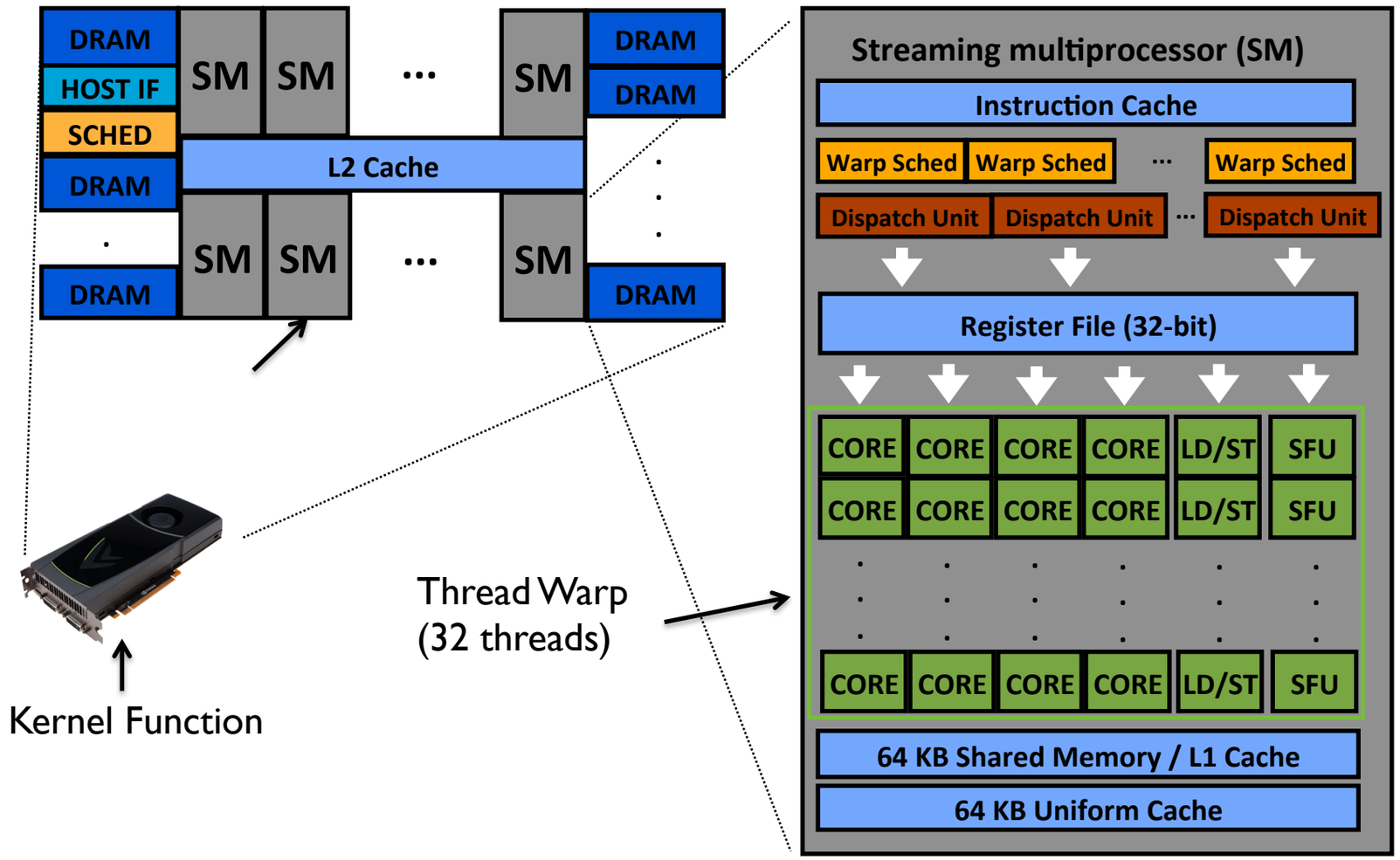
GOAL: FASTER NFS WITH GPUS?

- SIEVING: done on CPUs
- IDEA: offload **all post-sieving** to GPUs



GPUS, NOT ONLY GAMING…

- Massively parallel 32-bit many-core. Thousands of cores
- One int or float instruction/clock cycle per thread/core



INTEGER ARITHMETIC

	Fermi (GTX 500 family)	Kepler (GTX 700 family)	Maxwell (GTX 900 family)
Cores	Up to 512	Up to 2880	Up to 3072
SMs	Up to 16	Up to 15	Up to 24
Frequency	Up to 1544 Mhz	Up to 1100 Mhz (Max boost)	Up to 1200 Mhz (Max boost)
DRAM	Up to 3GB (192 GB/s)	Up to 6GB (336 GB/s)	Up to 12GB (336 GB/s)
Integer mul	**1/2 clock cycles**	1/6 clock cycles	Multiple instructions…

CUDA GPUS (NVIDIA)

- Tens of thousands of threads execute a GPU **kernel**
- Threads are grouped in **blocks** (at most 1024 threads).
  A **block** "goes" to a single SM and "resides" in it for whole execution
- **Warp**: batches of 32 threads within a **block** execute in lockstep on 32 cores
- Thread in a warp should not diverge and use coalesced memory access to get good performance…

POST-SIEVING ON GPUS

- **Input:** the set of pairs (a,b) output by the sieve, the coefficients of the two polynomials

- **Output:** Indices of pairs (a,b) that are relations (optionally found factors)

- **Two CUDA kernels run sequentially:**

1. Rational side: check $\mathbf{bf_r(a/b)}$ for B_L-smoothness (discard bad)

2. Algebraic side: check $\mathbf{b^d f_a(a/b)}$ for B_L-smoothness (output relations)

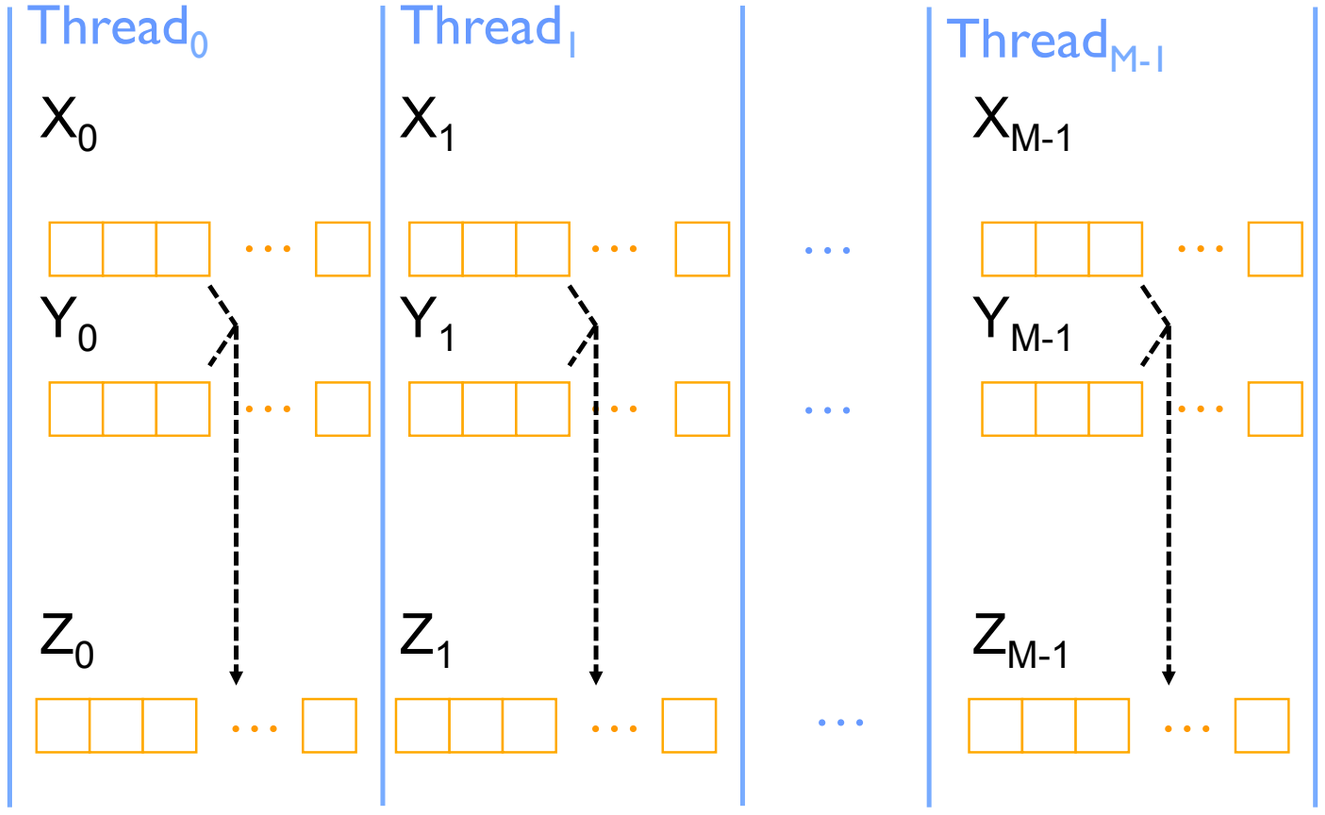
DESIGN STRATEGY

- Each thread processes one or more pairs (a,b) (task parallelism!)
- Each thread runs fixed sequence of steps to determine if value is B_r (B_a)-smooth except at most 3 (4) primes less than B_L

+ No thread synchronization, high computing/mem access ratio
- High register usage (and memory spilling…), high latency

ARITHMETIC DESIGN

- Sequential Radix 2^{32} Montgomery arithmetic
- PTX level optimized code (heavy use of MAD instructions!)

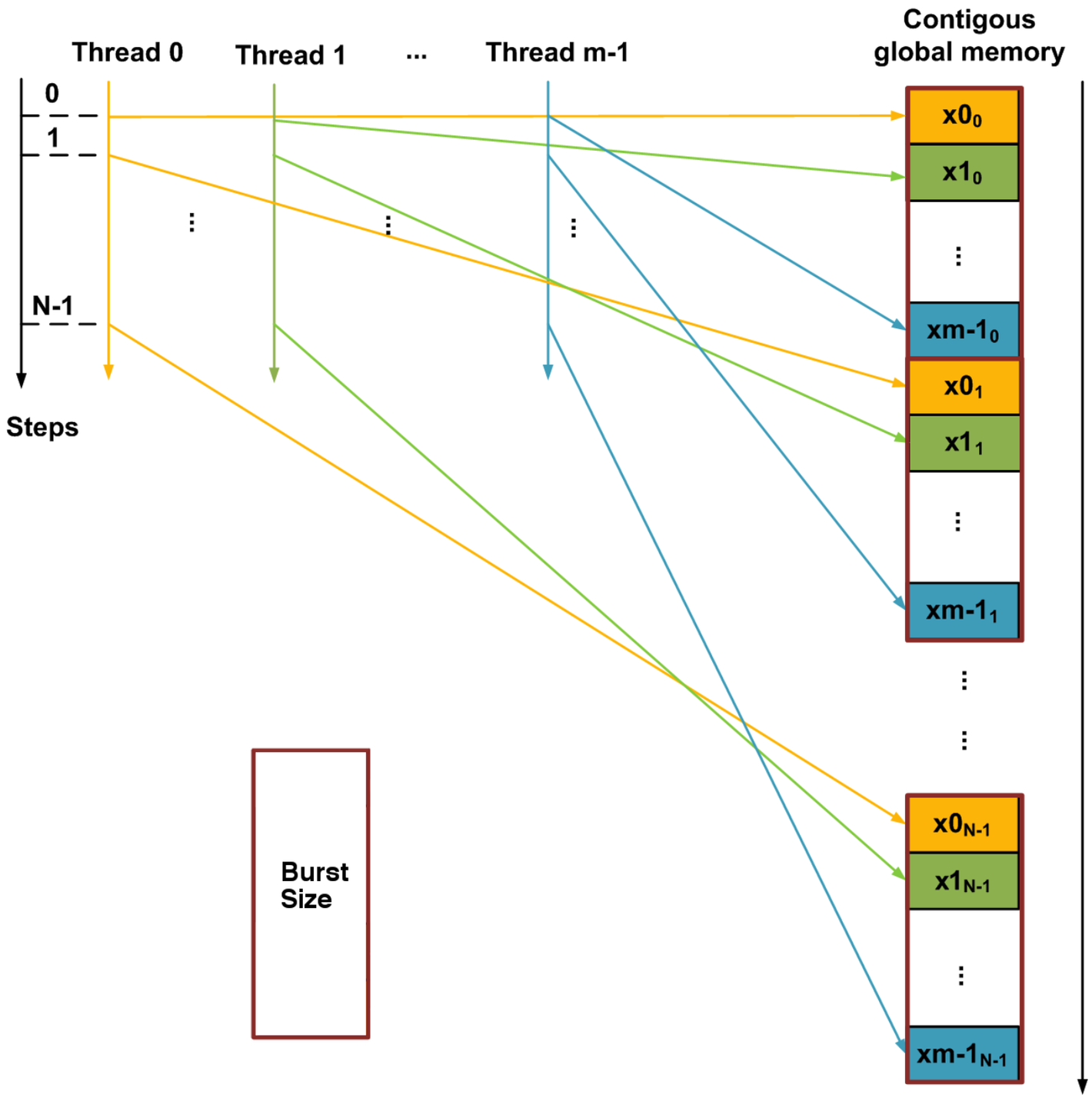


32-bit unsigned integer: □

Multi-precision integer: □ □ □ ··· □

$Z_i = OP(X_i, Y_i)$

COALESCED ACCESS



KERNEL DETAILS

- Rational side first (discard bad), then algebraic side (output relations)

A. Get pair **(a,b)** and evaluate polynomial $\mathbf{b \cdot f_{ra}(a/b)}$ (or $\mathbf{b^d \cdot f_{al}(a/b)}$)

B. Remove small factors: trial division, from now work on records: **(value, index)**

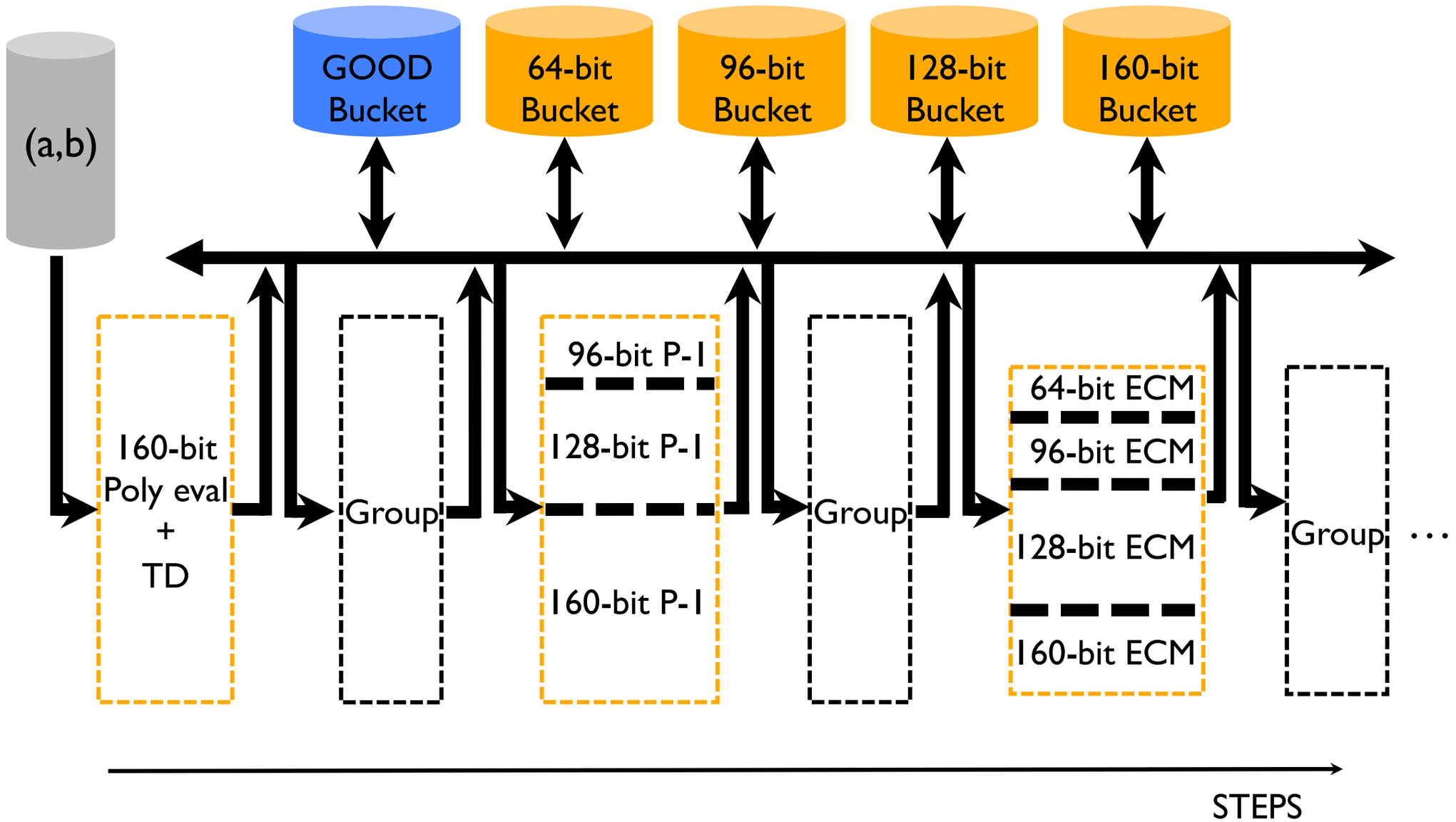
Repeat:

1. Group records in "buckets" according to value size

2. Factoring attempt: Pollard p-1 or ECM, if factor found, divide out

3. Test compositeness, discard prime values $> \mathbf{B_L}$, put aside values $\leq \mathbf{B_L}$

KERNEL WORKFLOW



ALGORITHMS

- **Bivariate polynomial evaluation:**
  naive, no Horner

- **Trial Division:**
  prime table in CMEM, divisibility test (Horner/Montgomery), exact div

- **Pseudo primality test (Montgomery arithmetic):**
  Selfridge-Rabin-Miller

- **Pollard P-1 (Montgomery arithmetic):**
  left-to-right modular exponentiation for stage 1, optimized BSGS for stage 2

- **ECM (Montgomery arithmetic):**
  Twisted Edwards curves, add chains for stage 1, optimized BSGS for stage 2

INTEGRATION WITH RSA-768 SOFTWARE

Finding good parameters for GPU kernels is hard!

- Preliminary experiments: rule out bad configurations
- We have run many experiments on RSA-768 datasets

What to optimize for?

- We have fixed the yield, and looked for fastest configurations
- Focus on two cases: 95% and 99% yield

	# Runs	Bounds (vals < 2^{256} , $B_L = 2^{37}$)
Trial Division	0-1	$B \cong 2^{10}$
Pollard p-1	1	$B_1 \cong 2^{10}$, $B_2 \cong 2^{14}$
ECM	8-20	$B_1=[2^8, 2^{10}]$, $B_2=[2^{12}, 2^{15}]$

RSA-768: CPU VS GPU

CPU: INTEL I7-3770K 4 cores 3.5 GHz 16GB RAM

Large primes	Input pairs	Tot time	Sieve time	PS-cof time	Relations found
≤ 3	≈ 5×10^5	29.6s	25.6s	4.0s	125 (31.3 rels/sec)
≤ 4	≈ 10^6	32.0s	25.9s	6.1s	137 (22.5 rels/sec)

GPU: NVIDIA GTX 580 512 CORES 1544 MHz 1.5 GB RAM

Large primes	Input pairs	Desired yield	CPU/GPU	Time	Relations found
≤ 3	≈ 5×10^5	95%	9.8	2.6s	132 (50.8 rels/sec)
		99%	6.9	3.7s	136 (36.8 rels/sec)
≤ 4	≈ 10^6	95%	4.0	6.5s	159 (24.5 rels/sec)
		99%	2.7	9.6s	165 (17.2 rels/sec)

RSA-768: 1 CPU VS 1 CPU + 1 GPU

Large primes	# Input pairs	Setting	Total time	# Relations found	Relations/sec
≤ 3	≈ 5×10^7	No GPU	2961s	12523	4.23
		With GPU	2564s	13761	5.37
≤ 4	≈ 5×10^7	No GPU	1602s	6855	4.28
		With GPU	1300s	8302	6.39

- Large primes ≤ 3: 24% GAIN
- Large primes ≤ 4: 45% GAIN

CONCLUSION

- GPUs are (were?) good accelerators for post sieving
- Their use can reduce overall NFS factoring time
- But taking into account power consumption, programming cost and reliability things get uglier…

THANKS FOR YOUR ATTENTION!