Numbers related to NFS

Kazumaro Aoki

NTT

Background

- NFS requires large amount of parameters. Though the order of the parameters are known, but how to set optimal number to the parameters are something art.
- Few data are public for large IF.
- Experimentalist (including me) are helped by seeing the numbers.

I hope that the numbers accelerates future factorizations.

Motivation and experience (1)

- During AES contest (1997-2000), I learned efficient software implementation.
- After AES, I implemented the fastest ECC implementation. (around 2000)
- I expect that I can get the world record of factorization soon.

I started to learn NFS with Prof. Kida and Dr. Shimoyama, and I soon realized that my expectation is very optimistic.

Motivation and experience (2)

- **Experimental and experienced data is very useful for experimentalist, but there is not so many way to publish except WWW.**
- **I announced our result in Japanese local technical report or workshops.**
- **This talk provides part of these results.**

Contents

- c176 in 11,281+ (2005 workshop on cryptography and related mathematics)
- Hybrid special-q
- Hardware failure and small tricks
- R311 ECM
- A little comments for memory usage

English publications

- **Asiacrypt 2004 — bucket sieve**
- **RSA100-150 in IACR ePrint archive**
- **2005 workshop on cryptography and related mathematics (talk only)**
  (http://www2.chuo-u.ac.jp/21COE/event/20050808-0810/200508.htm**)**
- **CHES06 — unluck (slides only)**
- **IWSEC07 — LA and c176**
- **Asiacrypt 2007 — M1039**

c176 in 11,281+

- My first world record of GNFS.
- The record was only kept in a week.
- RSA-200 factoring was announced May 2005.
- Technics were presented IWSEC2007.
- Parameters were already announced at “2005 Workshop on Cryptography and Related Mathematics” and web page.
- Following slides quotes the numbers from the above workshop and not included in IWSEC2007 paper.

of PCs Used in Each Step

	Step	distributed computing	# of PCs for GNFS176
1	poly. sel.	easy	52
2	sieve	easy	400
3	filtering	rel. easy	2
4	linear alg.	tight conn.	36
5	square root	rel. easy	36

Details of Our Program Running

	time spent GNFS176	
poly. sel.	20d	pol51m0b → pol51opt
	2h	mkprime
sieve	27d	ltsieve
filtering	4h	classifyRel → uniqRel, 32to64
	3h	getLP → countLP → lptxt2bin
	2h	sfctr
	8h	scmpi
	1h	compff → mkprematrixbin
	2d	splitpm + smerge
lin. alg.	1h	shufflematrix → mkmatrixbin
	1h	cut224mat → splitmatrix
	5d	planczos256
	1h	solve224mat → rff → gaussext
$\sqrt{}$	1h	anneal
	1h	papprox
	1h	pcouveignes, rsqrt

Program Lines

Step	# of lines	ratio
polynomial selection	5626	10%
sieve	16943	30%
filtering	17607	32%
linear algebra	7352	13%
square root	8150	15%
total	55678	100%

as of October 2005

Polynomial (1: statistics)

- Only one PC was used for first 2 weeks. (We were learning how to use the program.)
- On average 47 PCs were used during last 4 weeks, under the control of our parameter survey program.
- In total, we spent about 3.5 P4[3.2GHz] years.

characteristics of the output:

skewness 330579.16, norm 1.36e+24,
alpha -6.03, Murphy_E 1.46e-13

Polynomial (2: parameters)

pol51m0b

nprimes	normmax	a5(e-9) range
7	1e25	10-12, 20-80
8	1e26	0-10
	5e25	10-186, 190-206, 210-213, 220-223

pol51opt

normmax1	normmax2	murphye
1e25	1e22	1.0e-13

Sieving (1: time and hardware)

- 27 calendar days = 9.7 P4[3.2GHz] years.
- CPUs were PIII, P4, Athlon 64, and Opteron located at Rikkyo Univ, NTT, and Fujitsu Labs. Clock frequency ranged 1.0GHz-3.8GHz, amount of main RAM ranged 0.5GB-4GB. (speding 15.2 raw CPU years.)

Sieving (2: parameters)

- $512 \cdot 2^{20}$ **(=** $32 \cdot 2^{10} \times 16 \cdot 2^{10}$**) points were sieved for each special-q, and sieving rectangle was dependent on the lattice basis. (**$\approx$**3% improvement)**
- **special-q range: 28.8M-227.0M [10.5M ideals] (not included 111.5M-113.4M, and 17% of 28.8M-30.5M, 221.2M-227.0M)**
- **If we accept large matrix, we can reduce the above special-q range.**

Sieving (3: parameters cont'd)

- **factor base bound:** $0.95Q$ **for algebraic side, 80M for rational side.**
  (first 5.3M special-q for 512MB RAM PC)
- **large prime: bound is 4G, and 2 large primes are accepted for both sides.**
- **The base of** $\log$ **is scaled to fit 8-bit. The different scale was used for algebraic and rational sides.**
- **The parameters are NOT optimized. There may exist better bounds.**

Sieving (4: prime classification)

- **Pattern sieving is used for tiny primes ($3 \cdot 5 \cdot 7 \cdot 8$).**
- **Block sieving (we call it local sieve) is used for smallish primes.**
  [$\leq$ 2$\times$(width of sieving region)]
- **Bucket sorting (global sieve) is used for largish primes.**
- **Mini-primality testing and mini-factoring are used for large primes. (cofactorization)**

Sieving (5: sieving order)

1. algebraic global sieve
2. algebraic local sieve
3. rational local sieve
4. rational global trial sieve
5. rational trial division w/ mini-primality testing
6. algebraic trial division w/ mini-primality testing
7. mini-factoring for both sides

Sieving (6: global sieve)

- **Classical method (investigated by Pollard) was used for lattice points traveling.**
- **based on the bucket sorting algorithm for largish prime (Asiacrypt '04).**
  "Franke's siever" implements similar algorithm.
- **We stored both prime and $\log$ value.**
- **The contents in the buckets were reused for trial sieve.**

Sieving (7: local sieve)

- **Classical line-by-line sieving for $\log$ value.**
- **Sieving region was divided into the pieces whose size $\approx$ L2 cache.**
- **Prime ($\leq 2\times$ sieving region) powers ($\leq$ 320k) were also considered. (a few % improvement)**

Sieving (8: trial division)

- Trial sieving [GLM94] was used for largish prime.
- Large cofactor of norms was examined by mini-primality testing.
- Mini-factoring was applied if both algebraic and rational cofactors were composite.

Sieving (9: mini-primality testing)

- **Solovay-Strassen test with base 2.**
- **Optimized for two words (64-bit).**
- **Dussé-Kaliski version (word-wise) of Montgomery reduction, and 3-bit window method was used for exponentiation.**
- **Slightly improved version of Zassenhaus acceleration of Hensel lifting was used for computing $-n^{-1} \bmod 2^{32}$.**

Sieving (10: mini-factoring)

- **Optimized for two words (64-bit).**
- **$\approx 10\%$ were factored by $p-1$ method.**
- **SQUFOF was used for the input $< 2^{62}$ based on the Riesel book. We optimized the multiplier and the number of continued fractions. Moreover, we optimized the division in continued fraction, and square decision.**
- **ρ was used for $< 2^{64}$. Dussé-Kaliski version of Montgomery reduction was used.**

Filtering (1: procedure)

1. Duplicate relation removing and relation format examination.
2. Indexing large prime.
   (generation of free relations.)
3. Complete factorization of relations.
4. Removing singleton and clique relation.
5. Merge (=clever Gaussian reduction).

Filtering (2: duplicate removing)

1. **Format examination.**
2. **Relations were classified into 256 files indexed by** $\left\lfloor \frac{b}{8} \right\rfloor \bmod 256$**.**
3. **For each classified file, uniqueness was confirmed by an associated array in Perl.**

All steps can be parallelized, but only 2 PCs were used.

Filtering (2: indexing large prime)

- **Extract large primes from relation file and classified them by $\left\lfloor \frac{p}{2^{25}} \right\rfloor$.**
- **“`sort -u`” for each classified file.**
- **Discard Euler pseudoprimes.**
- **Free relations are generated by “extended” factor base. (not used)**

All steps can be parallelized, but only 2 PCs were used.

Filtering (3: complete factorization)

- Assign index for the factors in the relations.
- Examine that the relations are actually smooth.

All steps can be parallelized. We used 32 PCs.

Problem:

- Total size of the factor base file $\approx$ 4GB. (cannot expand to main RAM.)
- Trial division is moderately slow. ("hint" was included in the output of siever.)

Filtering (4: singleton-clique)

- Based on the Cavallar's algorithm. We did not consider free relations as special.
- Repeat "removing singleton and clique relations".
- A singleton step is 2-pass, and a clique step is 3-pass.
- 9 PCs (1 server + 8 clients) were used with MPI/GbE.

Filtering (5: merge)

- **Based on the Cavallar's algorithm. We did not consider free relations as special, and did not use "controlled-merge".**
- **3-pass on a PC with a SATA150 disk.**
- **We performed merge steps up to 20-way.**

k-way merge means that removes a factor and output $(k-1)$ relation-sets, when a factor is used k relation-sets.

Filtering (6: statistics)

576 372 161	**# of sieve outputs**
455 989 949	**# of unique relations**
328 707 916	**# of effective factor bases**
27 220 932	**# of relations after s-c step**
27 219 940	**# of factor bases**

output of merge:

$8\,526\,749 \times 8\,525\,751$ **(weight** $1\,707\,545\,745$**)**

Linear algebra (1: outline)

1. Removing heaviest 224 rows. Total weight was reduced from 1 707 545 745 to 1 394 050 132. (18% reduction)
2. Block Lanczos method for parallel processing. Block length was 256 bits. We got 256 block Lanczos solutions.
3. Recovery of heavy rows. # of solutions was reduced to 32.
4. Adjustment for quadratic characters. # of solutions was reduced to 24.

Linear algebra (2: performance)

- **36 P4s (3.2GHz$\times$32, 2.8GHz$\times$2, 3.6GHz$\times$2) with GbE found solutions in 5.3 days.**
- $(1/6) \times (1/6)$ **submatrix were assigned for each PC and balanced their weight.**
- $\approx$ **400MB RAM was required for each PC.**

Details of computation time	
matrix multiplication	33%
synchronization	26%
communication	21%
inner product	20%

Square root

- Based on Nguyen's method for algebraic side.
- Omitting ideal factorization for exceptional primes.
- Error term was computed by Couveignes method.
- 36 P4s were used.
- 1 hour was required for each solution.
  4th root factored c176 into p87 $\times$ p89.

Hardware failures in 3 years

40 servers including 32 2U P4[2.53GHz] servers.

- **15% HD were broken, but 90% were repaired by automatic reallocation of bad sectors.**
- **2 power units were broken.**
- **4 memory modules were broken.**
- **8 CPUs sometimes produced incorrect result.**
- **2 CPU fans were stopped.**
- **1 motherboard was broken.**
- **1 of 4 HUBs was broken.**

Hardware failures in 2.5 years

113 servers of 1U PD[3.0GHz].

- **68% HD were broken, but 97% were repaired by automatic reallocation of bad sectors.**
- **15% (67 out of 452) memory modules were broken. 48 memory modules were broken in last half year.**
- **3 motherboards were broken.**

Tricks for NFS

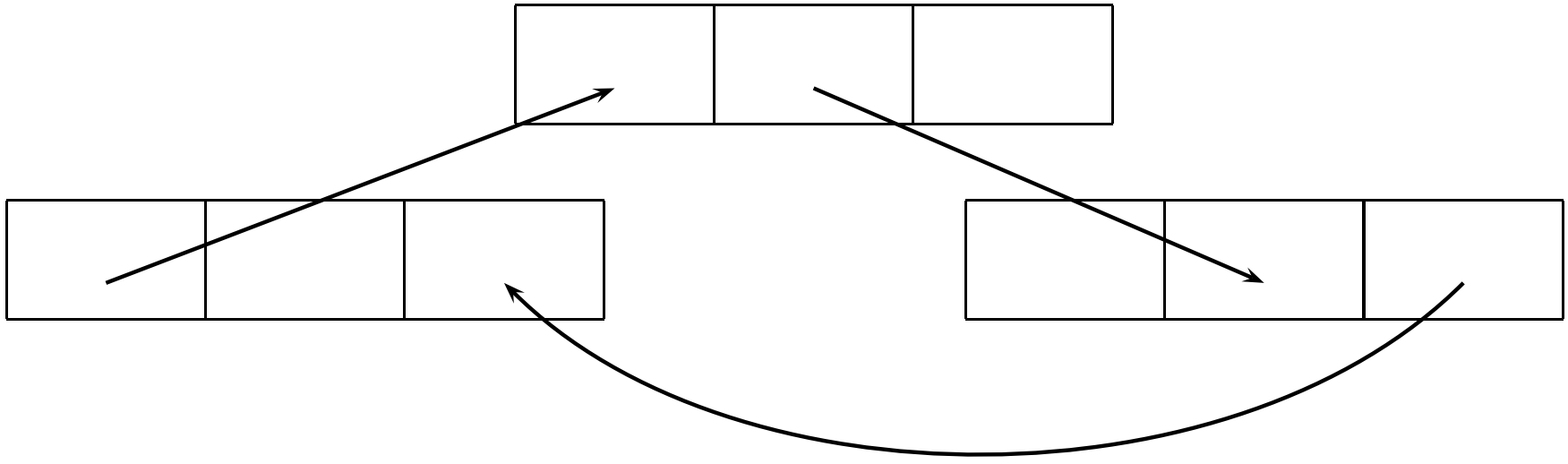
- Sum share algorithm for linear algebra step (reinvention of wheel?)
- Network construction for PC cluster (reinvention of wheel?)
- mini-primarity testing, SQUFOF
- lattice sieve

Sum Sharing

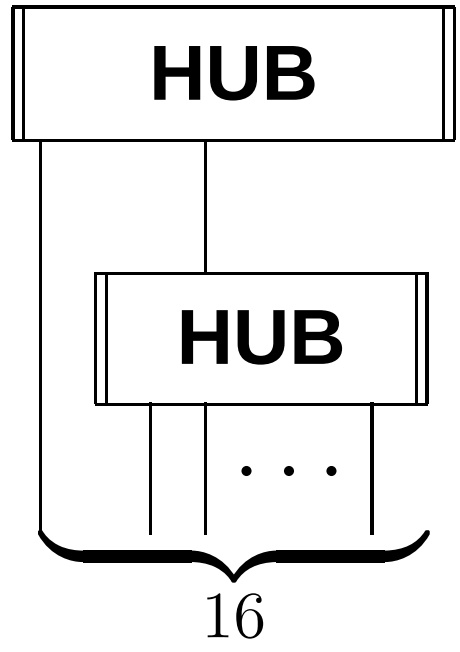
before: length l vector in n nodes

after: sum of all vectors shared in all nodes

A full-duplex ring network can realize in $2(n-1)\left\lceil \frac{l}{n} \right\rceil$, where length 1 vector can transfer in time 1.



Network Construction: 16 nodes



with 16-port HUB. each node has 1 NIC.

Network Construction: 36 nodes

1-	1-	12	12	13	13
1-	1-	12	12	13	13
12	12	2-	2-	23	23
12	12	2-	2-	23	23
13	13	23	23	3-	3-
13	13	23	23	3-	3-

using 3 20-port HUBs. each node has 2 NICs.

Base-2 Solovay-Strassen

- **Use of Montgomery reduction**
  - $(-n)^{-1} \bmod 2^{32}$**: HL improved by Zassenhaus**
  - **"1" in Montgomery domain:** $O \leftarrow (-n) \bmod 2^{64}$**;** $O \leftarrow O - \lfloor O/n \rfloor\, n$
- **3-bit window method, twice done by addition**
- **Jacobi symbol:**

$$n \equiv \begin{cases} \pm 1 \pmod 8 \Rightarrow \textbf{compare with } O \\ \pm 3 \pmod 8 \Rightarrow \textbf{compare with } n - O \end{cases}$$

Speeding up SQUFOF

- kn **factorization instead of** n **(**$\approx$ **10% improvement)**
- **Division by comparision because most of quotient is small (**$\approx$ **50% improvement)**
- **Square decision by** $\mathrm{mod}\, 2^8$ **and** $\mathrm{mod}\,(2^8 - 1)$ **(**$\approx$ **83% improvement)**
- **least 15 bits of square root of 32-bit numbers uniquely determined by upper 17 bits.**

Speeding up lattice sieve

- **binary GCD: updating variables are assigned to an array. conditional branch can be removed.**
- **don't sieve** $\gcd(c, d) = 2$ **in** $[c, d]$**-plane.**
- **range of** $[c, d]$**-plane is adaptively changed by the special-**q **basis (**$\approx$ **3% improvement)**

On hybrid special-q

- There is a peak efficiency (yield/second). When $q <$peak, then the efficiency quickly droppped. When $q >$peak, then the efficiency is smoothly dropped.
- On SNFS, the norm of rational side and algebraic side are not so different. Applying special-q on each side has similar factorization time.
- How about applying special-q on both side? (cf. M809 example by Franke)

Example: SNFS248

$$\begin{aligned} N &= 2^{821} + 2^{411} + 1 \\ F(a,b) &= |a^6 + 2a^3(-b)^3 + 2(-b)^6| \\ G(a,b) &= |a + 2^{137}b| \end{aligned}$$

$|a|, |b| \leq H \approx 2^{27}$, **therefore**

$$\begin{aligned} \log_2 F(a,b) &\approx 27 \times 6 = 162 \\ \log_2 G(a,b) &\approx 27 + 137 = 164 \end{aligned}$$

special-q is set to $q \mid G$.

Sieving result

rational side (G)										
region	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9
# of rel (M)	65	66	63	61	56	56	51	48	50	48

algebraic side (F)			
region	A1	A2	A3
# of rel (M)	65	59	56

We used to most of R0$\sim$R9 to factor.

558M relation $\longrightarrow$ 438M$\times$324M matrix (dup 21.5%)

How many relations are required to factor?

region	yield	dup	matrix size	s-c
R0123456	418M	21%	329M×297M	30M
R1234567	401M	17%	331M×297M	31M
R012345	367M	20%	292M×285M	35M
R123456	353M	16%	295M×286M	35M
R01234	311M	19%	251M×269M	46M
R12345	302M	15%	256M×271M	45M
R0123	255M	18%	208M×248M	—
R1234	246M	13%	213M×250M	—

Use of hybrid special-q

region	yield	dup
A123R01	312M	29%
A123R12	310M	26%
A12R012	318M	31%
A12R123	315M	27%
R12345	302M	15%

- Hybrid special-q generates many duplicated relations.
- We tried other possible combinations and the results seem similar.

ECM factored R311

- **R311 =** $(10^{311} - 1)/9$
- **At that time, the 2nd largest record by ECM, and only repunit in Cunnigham table with no known factor.**

Standard PC

CPU: Pentium 4 [3.2GHz] (Northwood) HT

RAM: 2GB

OS: FreeBSD 4.11R

We can use 32 set of above PC.

Parameter selection

Rule of thumb: to find $\frac{2}{9}$-digit factor

This time: $311 \cdot \frac{2}{9} \approx 69$

To set the success probability with 63%

default value in GMP-ECM6 for 65-digit factor:

$B_1 = 850 \times 10^6$

$\Downarrow$

more than 28 years are required for one standard PC very difficult

Small B_1 may miss a large factor

$\rightarrow$ **choose $B_1 = 850 \times 10^6$ (B_2 is optimized for 2GB RAM)**

Basic strategy for distributed computing

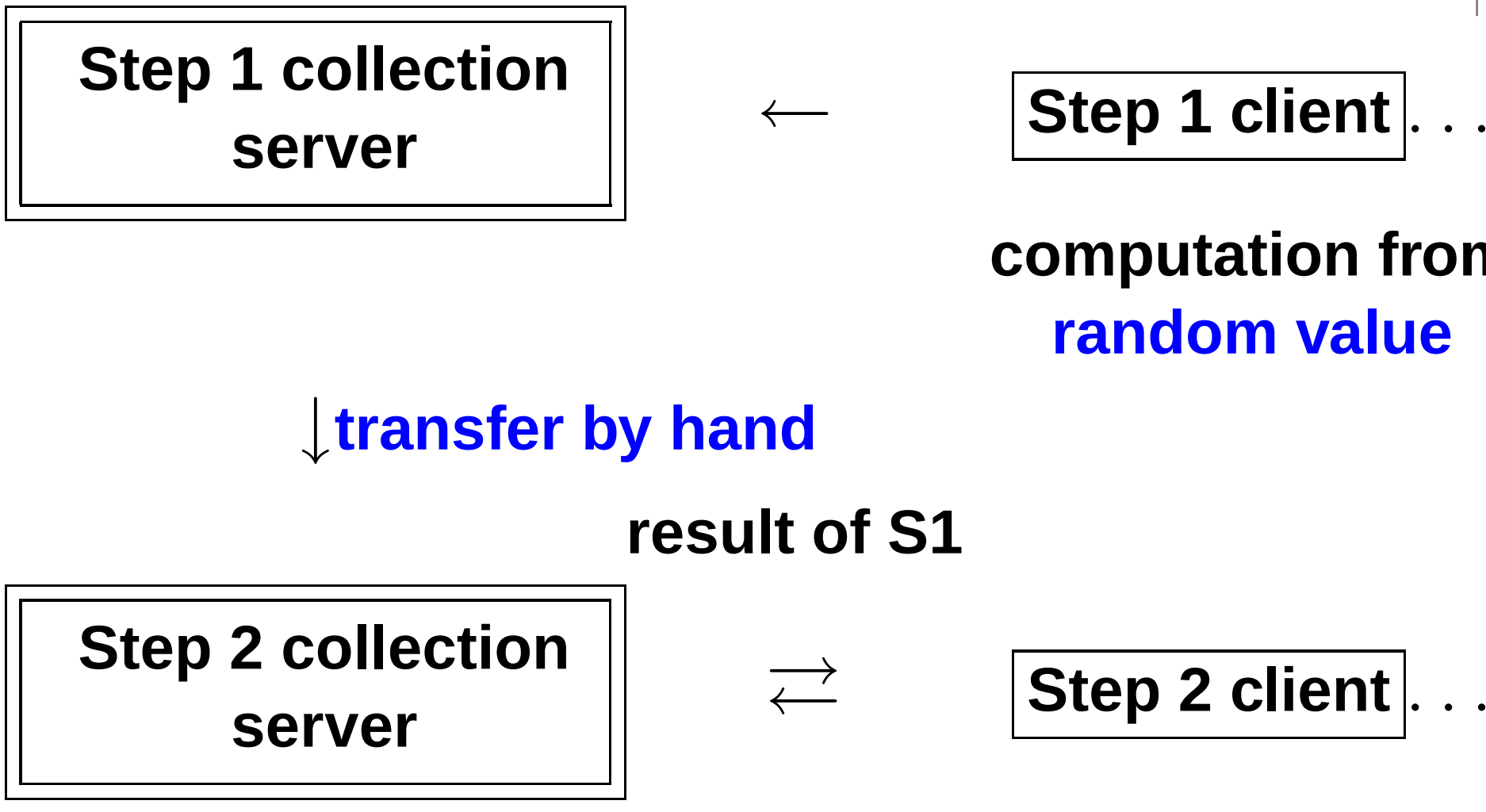
Characteristics of ECM implementation:

Step 1: ALU (especially multiplier) is required

Step 2: speeding up by large amount of memory

- **Step 1 is computed by volunteering idle PCs**
- **Step 2 is computed by occupiable PCs**

Structure



Step 2 client

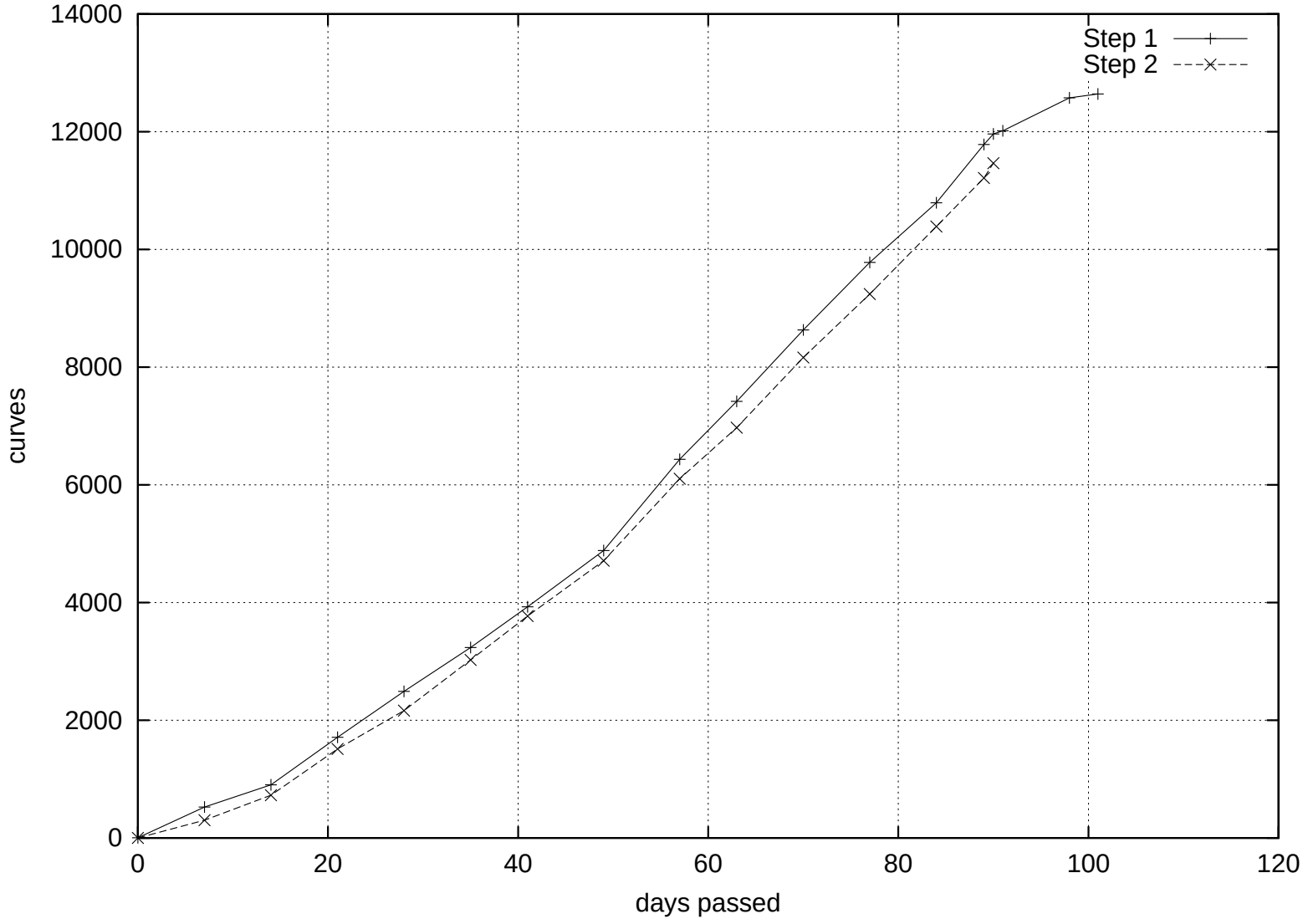
- Standard PC×32 (gradually increasing)

  We used two sets of following PCs.
- Pentium 4 Xeon NW 2.8GHz + 2GB RAM
- Pentium 4 PS EM64T 3.6GHz + 2GB RAM
- Athlon 64 2.2GHz + 3GB RAM
- Opteron 2.0GHz + 4GB RAM

Step 1 client software

OS	CPU
FreeBSD 4.x, 5.x	Pentium 4
Linux	Pentium 4
FreeBSD 4.x, 5.x	AMD64 and EM64T
Linux	AMD64 and EM64T
Darwin	PowerG5
Linux	Itanium 2
FreeBSD 4.x, 5.x	P6
FreeBSD 4.x, 5.x	K6-2
Microsoft Windows	Pentium 4, Pentium D
Microsoft Windows	P6

of curves, days passed



Comments

- Call volunteering PCs for two MLs in NTT (# of recipient is about 100)
- Executable for Microsoft Windows are provided (14th day acceleration)
- Executable with AMD64 patch are provied (47th day acceleration)
- found a factor at 89th day

Statistics 1

# of curves when factor is found

Step 1 11784 curves

Step 2 11214 curves

Size of expected factor	missing probability
55 digits	2.0% at most
60 digits	43.5% at most
65 digits	85.0% at most

Statistics 2

Applying the fastest PC (Opteron 2.0GHz + 4GB RAM) we used:

Step 1	**4.4 hours**
Step 2	**1.6 hours**

7.91 PC (Opteron 2.0GHz + 4GB RAM)●year

Statistics 3

# of IP addresses detected at Step 1	
volunteering PC in NTT	96
occupied PC	40

Ratio of contribution		
	total	only Step 1
volunteering PC in NTT	44%	60%
occupied PC	56%	40%

average ability for Step 1: 0.21 Opteron 2.0GHz

Final remark

- **Spread of 64-bit CPU is probably accelerates NFS effort.**
- **Memory requirement after sieving is probably cause problems.**

T: $L_N[1/3, (64/9)^{1/3}]$ **M:** $L_N[1/3, (8/9)^{1/3}]$

$$\Downarrow$$

Time: $L_N[1-2s]$ **Memory:** $L_N[s]\ (s \leq 1/3)$

Photos: If time remains.

