What's the best data structure for multivariate polynomials in a world of 64 bit multicore computers?

Michael Monagan

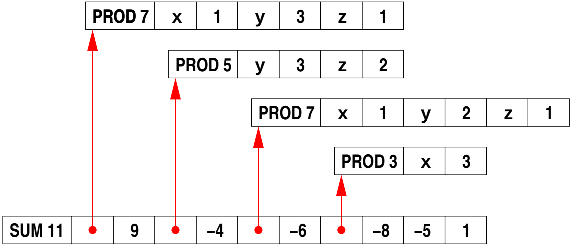
Center for Experimental and Constructive Mathematics
Simon Fraser University
British Columbia

ECCAD 2013, Annapolis, Maryland
April 27, 2012

This is joint work with Roman Pearce.

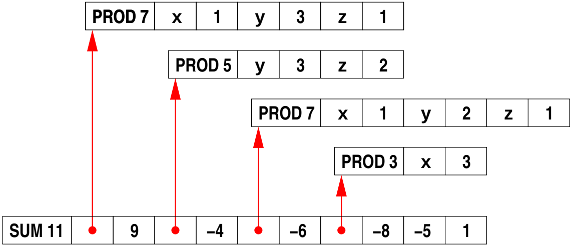
Representations for $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5$.

Maple 16

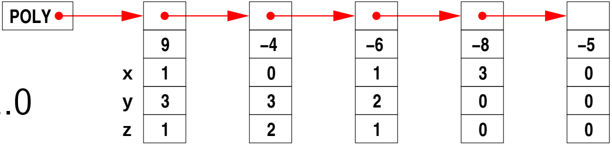


Representations for $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5.$

Maple 16



Singular 3.1.0



- Memory access is not sequential.
- Monomial multiplication costs O(100) cycles.

Our representation $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5$.

SEQ 4	x	y	z

POLY 12	•	5131	9	5032	–4	4121	–6	3300	–8	0000	–5

Monomial encoding for graded lex order with $x > y > z$

Monomial $>$ and $\times$ cost **one** instruction !!!!

Advantages

Our representation $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5$.

SEQ 4	x	y	z

POLY 12	•	5131	9	5032	–4	4121	–6	3300	–8	0000	–5

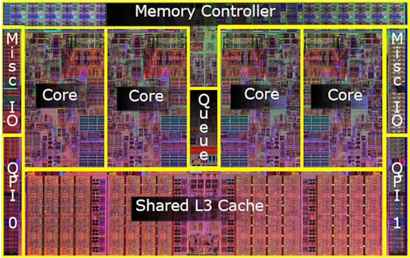
Monomial encoding for graded lex order with $x>y>z$

Monomial $>$ and $\times$ cost **one** instruction !!!!

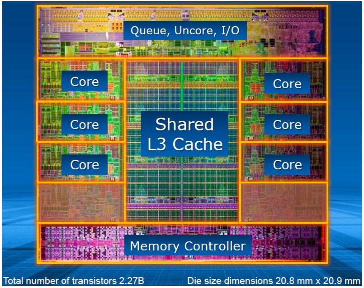
Advantages

- It's about four times more compact.
- Memory access is sequential.
- The simpl table is not filled with PRODs.
- Division cannot cause exponent overflow in a graded lex order.

Multicore Computers: Intel's Corei7

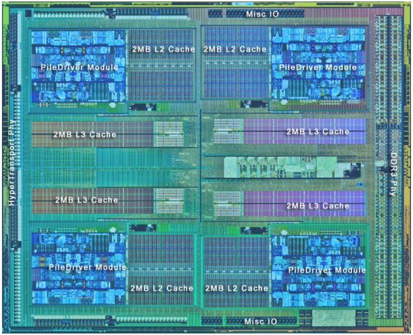


Core i7 920 @ 2.67 GHz
45nm lithography, Q4 2008



Core i7-3930K @ 3.20 GHz
32 nm lithography, Q4 2011
Overclocked @ 4.2 GHz

Multicore Computers: AMD FX 8350 Intel i7 4770



AMD FX 8350 @ 4.2 GHz
8 core, 32nm, Q4, 2012
Full integer support.

Intel Core i7-4770 @ 3.5 GHz
4 core, 22 nm, Q2 2013
Only 5–10% faster.

How should we parallelize Maple?
How would that speed up polynomial factorization?

Talk Outline

Let's parallelize polynomial multiplication and division.

- Johnson's sequential polynomial multiplication
- Our parallel polynomial multiplication
- A multiplication and factorization benchmark

Why is parallel speedup poor?

- Maple 17 integration of POLY
- New timings for same benchmark.
- Notes on integration into Maple 17 kernel.
- Future work.

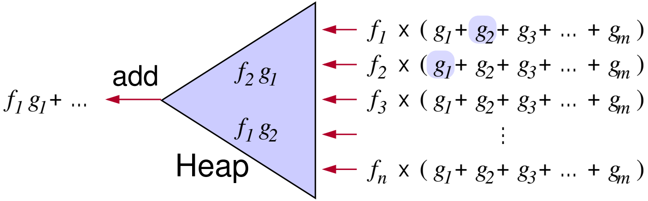
Sequential multiplication using a binary heap.

Let $f = f_{\mathbf{1}} + \cdots + f_{\mathbf{n}} = c_1 X_1 + \cdots c_n X_n$.
Let $g = g_{\mathbf{1}} + \cdots + g_{\mathbf{m}} = d_1 Y_1 + \cdots d_m Y_m$.
Compute $f \times g = f_1 \cdot g + f_2 \cdot g + \cdots + f_n \cdot g$.

Johnson (1974) simultaneous n-ary merge (heap): $O(mn \log n)$.

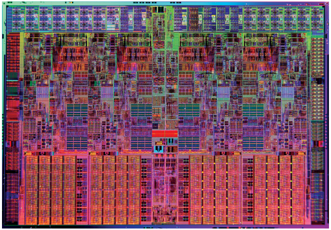
Let $f = f_{\mathbf{1}} + \cdots + f_{\mathbf{n}} = c_1 X_1 + \cdots c_n X_n$.
Let $g = g_{\mathbf{1}} + \cdots + g_{\mathbf{m}} = d_1 Y_1 + \cdots d_m Y_m$.
Compute $f \times g = f_1 \cdot g + f_2 \cdot g + \cdots + f_n \cdot g$.

Johnson (1974) simultaneous n-ary merge (heap): $O(mn \log n)$.

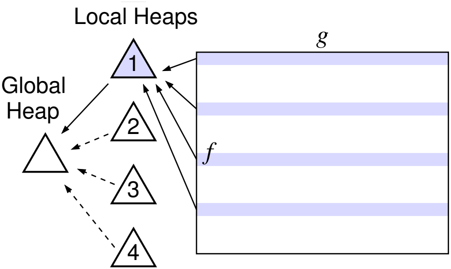


- $|Heap| \leq n \implies O(nm \log n)$ comparisons.
- Delay coefficient arithmetic to eliminate garbage!

Parallel multiplication using a binary heap.

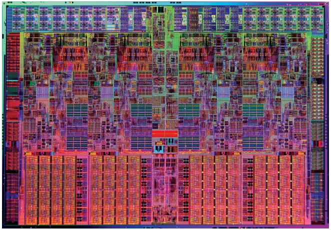


Target architecture

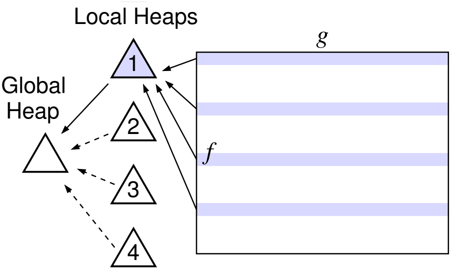


One thread per core.

Parallel multiplication using a binary heap.

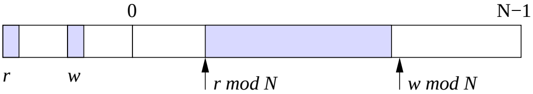


Target architecture



One thread per core.

Threads write to a finite circular buffer.



Threads try to acquire global heap as buffer fills up to balance load.

Maple 16 multiplication and factorization benchmark.

Intel Core i7 920 2.66 GHz (4 cores) Times in seconds

multiply	Maple 13	Maple 16 1 core	Maple 16 4 cores	Magma 2.16-8	Singular 3.1.0	Mathematica 7
$p_1 := f_1(f_1+1)$	1.60	0.063	0.030	0.30	0.58	4.79
$p_4 := f_4(f_4+1)$	95.97	2.14	0.643	13.25	30.64	273.01
factor	**Hensel lifting is mostly polynomial multiplication!**					
p_1 12341 terms	31.10	2.80	2.65	6.15	12.28	11.82
p_4 135751 terms	2953.54	59.29	46.41	332.86	404.86	655.49

$f_1 = (1+x+y+z)^{20}+1$ 1771 terms

$f_4 = (1+x+y+z+t)^{20}+1$ 10626 terms

Parallel speedup for $f_4 \times (f_4+1)$ is 2.14 / .643 = **3.33**×. **Why?**

Maple 16 Integration of POLY

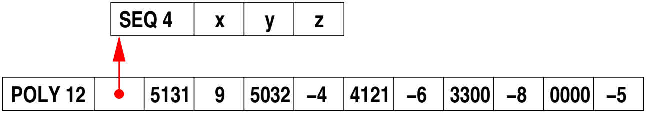
To expand sums $f \times g$ Maple calls `` `expand/bigprod(f,g)` ``
if $\#f > 2$ and $\#g > 2$ and $\#f \times \#g > 1500$.

```
`expand/bigprod` := proc(a,b)  # multiply two large sums
   if type(a,polynom(integer)) and type(b,polynom(integer)) then
     x := indets(a) union indets(b); k := nops(x);
     A := sdmp:-Import(a, plex(op(x)), pack=k);
     B := sdmp:-Import(b, plex(op(x)), pack=k);
     C := sdmp:-Multiply(A,B);
     return sdmp:-Export(C);
   else
   ...

`expand/bigdiv` := proc(a,b,q)  # divide two large sums
   ...
     x := indets(a) union indets(b); k := nops(x)+1;
     A := sdmp:-Import(a, grlex(op(x)), pack=k);
     B := sdmp:-Import(b, grlex(op(x)), pack=k);
   ...
```

Make POLY the default representation in Maple.

If we can pack all monomials into one word use



otherwise use the sum-of-products structure.

Make POLY the default representation in Maple.

If we can pack all monomials into one word use

SEQ 4	x	y	z

POLY 12	•	5131	9	5032	–4	4121	–6	3300	–8	0000	–5

otherwise use the sum-of-products structure.

But must reprogram entire Maple kernel for new POLY !!

$O(1)$	`degree(f); lcoeff(f); indets(f);`
$O(n+t)$	`degree(f,x); expand(x*t); diff(f,x);`

For f with t terms in n variables.

High performance solutions: **coeff**

SEQ 4	x	y	z

POLY 12	•	5131	9	5032	–4	4121	–6	3300	–8	0000	–5

To compute `coeff(f,y,3)` we need to

$$\boxed{d \mid i \mid 3 \mid k} \xrightarrow{1} \boxed{0 \mid d-3 \mid i \mid k} \xrightarrow{2} \boxed{d-3 \mid i \mid k}$$

We can do step 1 in $O(1)$ bit operations.
Can we do step 2 faster than $O(n)$ bit operations?

High performance solutions.

```
/* pre-compute masks for compress_fast */
static void compress_init(M_INT mask, M_INT *v)

/* compress monomial m using precomputed masks v */
/* in O( log_2 WORDSIZE ) bit operations */
static M_INT compress_fast(M_INT m, M_INT *v)
{     M_INT t;
      if (v[0]) t = m & v[0], m = m ^ t | (t >> 1);
      if (v[1]) t = m & v[1], m = m ^ t | (t >> 2);
      if (v[2]) t = m & v[2], m = m ^ t | (t >> 4);
      if (v[3]) t = m & v[3], m = m ^ t | (t >> 8);
      if (v[4]) t = m & v[4], m = m ^ t | (t >> 16);
#if WORDSIZE > 32
      if (v[5]) t = m & v[5], m = m ^ t | (t >> 32);
#endif
      return m;
}
```

- Costs 24 bit operations per monomial.
- Intel Haswell (2013): 1 cycle (PEXT/PDEP)

Result: everything except op and map is fast!

command	Maple 16	Maple 17	speedup	notes
coeff$(f, x, 20)$	2.140 s	0.005 s	420x	terms easy to locate
coeffs(f, x)	0.979 s	0.119 s	8x	reorder exponents and radix
frontend$(g, [f])$	3.730 s	0.000 s	$\rightarrow O(n)$	looks at variables only
degree(f, x)	0.073 s	0.003 s	24x	stop early using monomial de
diff(f, x)	0.956 s	0.031 s	30x	terms remain sorted
eval$(f, x = 6)$	3.760 s	0.175 s	21x	use Horner form recursively
expand$(2 * x * f)$	1.190 s	0.066 s	18x	terms remain sorted
indets(f)	0.060 s	0.000 s	$\rightarrow O(1)$	first word in dag
op(f)	0.634 s	2.420 s	0.26x	has to construct old structur
for t in f do	0.646 s	2.460 s	0.26x	has to construct old structur
subs$(x = y, f)$	1.160 s	0.076 s	15x	combine exponents, sort, me
taylor$(f, x, 50)$	0.668 s	0.055 s	12x	get coefficients in one pass
type$(f, polynom)$	0.029 s	0.000 s	$\rightarrow O(n)$	type check variables only

For f with $n = 3$ variables and $t = 10^6$ terms created by

```
f := expand(mul(randpoly(v,degree=100,dense),v=[x,y,z])):
```

Maple 17 multiplication and factorization benchmark

Intel Core i5 750 2.66 GHz (4 cores) — Times in seconds

multiply	Maple 16 1 core	Maple 16 4 cores	Maple 17 1 core	Maple 17 4 cores	Magma 2.19-1	Singular 3.1.4
$p_4 := f_4(f_4+1)$	2.140	0.643	1.770	0.416	13.43	31.59
$p_6 := f_6 g_6$	0.733	0.602	0.203	0.082	0.90	2.75
factor	**Singular's factorization improved!**					
p_4 135751 terms	59.27	46.41	24.35	12.65	325.26	61.05
p_6 417311 terms	51.98	49.07	8.32	6.32	364.67	42.08

$$f_4 = (1+x+y+z+t)^{20}+1 \quad \text{10626 terms}$$
$$f_6 = (1+u^2+v+w^2+x-y)^{10}+1 \quad \text{3003 terms}$$
$$g_6 = (1+u+v^2+w+x^2+y)^{10}+1 \quad \text{3003 terms}$$

Parallel speedup for $f_4 \times (f_4+1)$ is $1.77/0.416 = \mathbf{4.2}\times$.

Notes on integration of POLY for Maple 17

Given a polynomial $f(x_1, x_2, ..., x_n)$, we store f using POLY if

(1) f is expanded and has integer coefficients,

(2) $d > 1$ and $t > 1$ where $d = \deg f$ and $t = \#\text{terms}$,

(3) we can pack all monomials of f into **one 64 bit word**, i.e. if $d < 2^b$ where $b = \lfloor \frac{64}{n+1} \rfloor$

Otherwise we use the sum-of-products representation.

Notes on integration of POLY for Maple 17

Given a polynomial $f(x_1, x_2, ..., x_n)$, we store f using POLY if

(1) f is expanded and has integer coefficients,

(2) $d > 1$ and $t > 1$ where $d = \deg f$ and $t = \#\text{terms}$,

(3) we can pack all monomials of f into **one 64 bit word**, i.e. if $d < 2^b$ where $b = \lfloor \frac{64}{n+1} \rfloor$

Otherwise we use the sum-of-products representation.

- The representation is invisible to the Maple user. Conversions are automatic.

Notes on integration of POLY for Maple 17

Given a polynomial $f(x_1, x_2, ..., x_n)$, we store f using POLY if

(1) f is expanded and has integer coefficients,

(2) $d > 1$ and $t > 1$ where $d = \deg f$ and $t = \#\text{terms}$,

(3) we can pack all monomials of f into **one 64 bit word**, i.e. if $d < 2^b$ where $b = \lfloor \frac{64}{n+1} \rfloor$

Otherwise we use the sum-of-products representation.

- The representation is invisible to the Maple user. Conversions are automatic.
- POLY polynomials will be displayed in sorted order.

Notes on integration of POLY for Maple 17

Given a polynomial $f(x_1, x_2, ..., x_n)$, we store f using POLY if

(1) f is expanded and has integer coefficients,

(2) $d > 1$ and $t > 1$ where $d = \deg f$ and $t = \#\text{terms}$,

(3) we can pack all monomials of f into **one 64 bit word**, i.e. if $d < 2^b$ where $b = \lfloor \frac{64}{n+1} \rfloor$

Otherwise we use the sum-of-products representation.

- The representation is invisible to the Maple user. Conversions are automatic.
- POLY polynomials will be displayed in sorted order.
- Packing is fixed by $n = \#\text{variables}$.

Degree limits (64 bit word)

	per variable		total degree	
n	#bits	max deg	extra bits	max deg
6	9	511	1	1023
7	8	255	0	255
8	7	127	1	255
9	6	63	4	1023
10	5	31	9	16383
11	5	31	4	511
12	4	15	12	65535
13	4	15	8	4095
14	4	15	4	255
15	4	15	0	15
16	3	7	13	65535
19	3	7	4	127
20	3	7	1	15

Joris van der Hoven: Do you use the extra bits for the total degree?
My answer: No, because it would complicate and slow down the code, e.g., polynomial division would require explicit overflow checking.
E.g. $b = 2$ $x^2y^2 + y^3 \div x^2y + y^3 = y$ with remainder $-y^4$.

Degree limits (64 bit word)

	per variable		total degree		Vandermonde	
n	#bits	max deg	extra bits	max deg	deg(det(V_n))	time(s)
6	9	511	1	1023	15	0.008s
7	8	255	0	255	21	0.008s
8	7	127	1	255	28	0.043s
9	6	63	4	1023	36	0.264s
10	5	31	9	16383	45	43.83s
11	5	31	4	511	55	–
12	4	15	12	65535	66	–
13	4	15	8	4095	78	–
14	4	15	4	255	91	–
15	4	15	0	15	–	–
16	3	7	13	65535	–	–
19	3	7	4	127	–	–
20	3	7	1	15	–	–

Joris van der Hoven: Do you use the extra bits for the total degree?
My answer: No, we can multiply $f \times g$ in POLY if $\deg f + \deg g < 2^b$.
Moreover, polynomial division would require explicit overflow checking.
E.g. $x^2y^2 + y^3 \div x^2y + y^3 = y$ with remainder y^4.

Future Work

- POLY is in Maple 17 !

Future Work

- POLY is in Maple 17 !
- Use extra bits for total degree.

Future Work

- POLY is in Maple 17 !
- Use extra bits for total degree.
- Rethink polynomial factorization for multi-core computers.

	factor(p)				p := expand(f×g)			
# cores	1	2	4	6	1	2	4	6
real time	97.51s	55.36s	36.85s	31.59s	5.60s	2.50s	1.18s	0.78s
speedup	–	1.8x	2.7x	3.1x	–	2.2x	4.7x	7.1x

Intel Core i7 3930K, **6 cores**, overclocked @ 4.2GHz

Future Work

- POLY is in Maple 17 !
- Use extra bits for total degree.
- Rethink polynomial factorization for multi-core computers.

	factor(p)				p := expand(f×g)			
# cores	1	2	4	6	1	2	4	6
real time	97.51s	55.36s	36.85s	31.59s	5.60s	2.50s	1.18s	0.78s
speedup	–	1.8x	2.7x	3.1x	–	2.2x	4.7x	7.1x

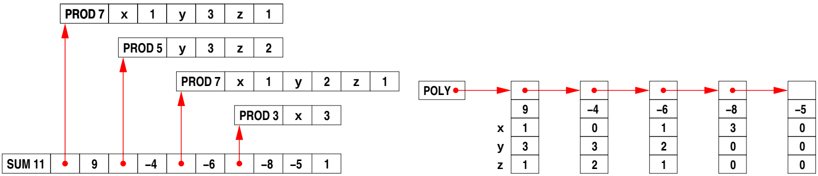
Intel Core i7 3930K, **6 cores**, overclocked @ 4.2GHz

Let $f(u,v,w,x,y) = \left(\sum c_{i,j}(u,v,w)x^i y^j\right) \times \left(\sum d_{i,j}(u,v,w)x^i y^j\right)$.
Pick $\alpha = (\omega_1, \omega_2, \omega_3) \in \mathbb{Z}_p^3$ and for $k = 1, 2, \cdots$ factor

$$f(\alpha^k, x, y) = \left(\sum c_{i,j}(\alpha^k)x^i y^j\right) \times \left(\sum d_{i,j}(\alpha^k)x^i y^j\right) \bmod p.$$

Conclusion

We will not get good parallel speedup using these



Even with conversions to a more suitable data structure, sequential overhead will limit parallel speedup.

Thank you for attending my talk.