POLY : A new polynomial data structure for Maple 17 that improves parallel speedup.

Michael Monagan

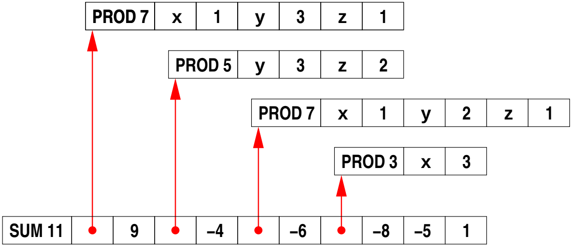
Centre for Experimental and Constructive Mathematics
Simon Fraser University.

Maplesoft presentation, August 14th, 2012

This is joint work with Roman Pearce.

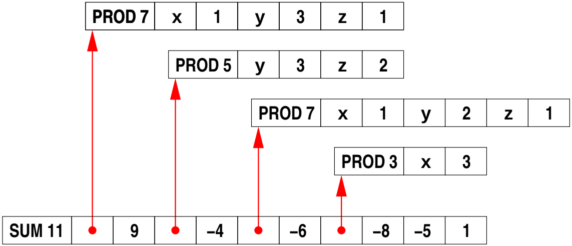
Representations for $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5$.

Maple 16

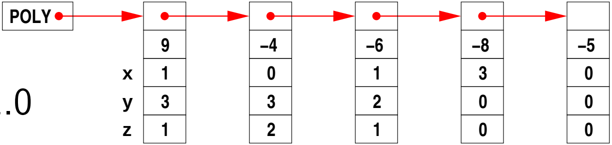


Representations for $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5$.

Maple 16

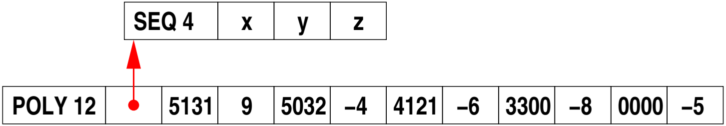


Singular 3.1.0



- Memory access is not sequential.
- Monomial multiplication costs circa 200 cycles.

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



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

Immediate 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 $x > y > z$

Immediate advantages:

- It's about four times more compact.

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 $x > y > z$

Immediate advantages:

- It's about four times more compact.
- Memory access is sequential.

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 $x>y>z$

Immediate advantages:

- It's about four times more compact.
- Memory access is sequential.
- The simpl table is not filled with `PROD`s.

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

SEQ 4	x	y	z

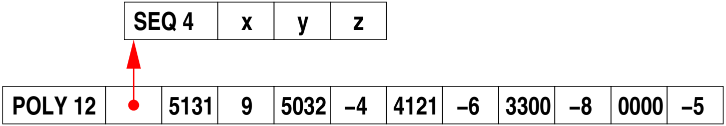
POLY 12	•	5131	9	5032	−4	4121	−6	3300	−8	0000	−5

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

Immediate advantages:

- It's about four times more compact.
- Memory access is sequential.
- The simpl table is not filled with PRODs.
- Monomial $>$ and $\times$ cost one instruction.

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



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

Immediate advantages:

- It's about four times more compact.
- Memory access is sequential.
- The simpl table is not filled with PRODs.
- Monomial $>$ and $\times$ cost one instruction.
- Division cannot cause exponent overflow in graded lex order.

Talk Outline

- Sequential polynomial multiplication
- Parallel polynomial multiplication
- A multiplication and factorization benchmark

Why is parallel speedup poor?

We've made POLY the default in Maple.

- New code
- New timings
- Integration details
- Reflections
- Future

Multiplication using a binary heap.

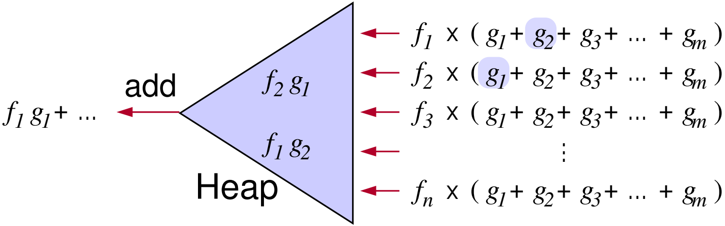
Let $f = f_1 + f_2 + \cdots + f_{\mathbf{n}}$ and $g = g_1 + g_2 + \cdots + g_{\mathbf{m}}$.
Compute $f \times g = f_1 \cdot g + f_2 \cdot g + \cdots + f_n \cdot g$.

Johnson, 1974, does a simultaneous *n*-ary merge using a heap.

Multiplication using a binary heap.

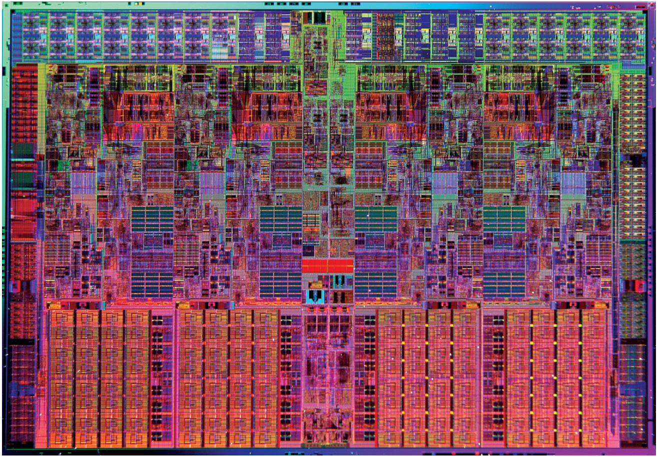
Let $f = f_1 + f_2 + \cdots + f_{\mathbf{n}}$ and $g = g_1 + g_2 + \cdots + g_{\mathbf{m}}$.
Compute $f \times g = f_1 \cdot g + f_2 \cdot g + \cdots + f_n \cdot g$.

Johnson, 1974, does a simultaneous n-ary merge using a heap.



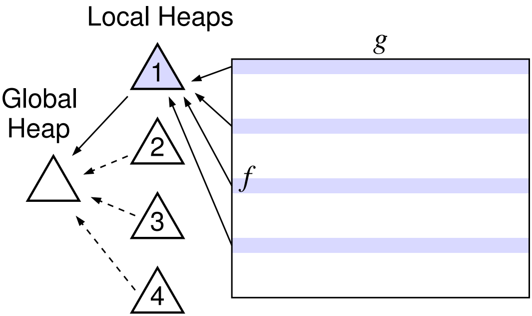
- $|Heap| \leq n \implies O(nm \log n)$ comparisons.
- Implementation uses $O(n + k)$ working space.

Target Parallel Architecture



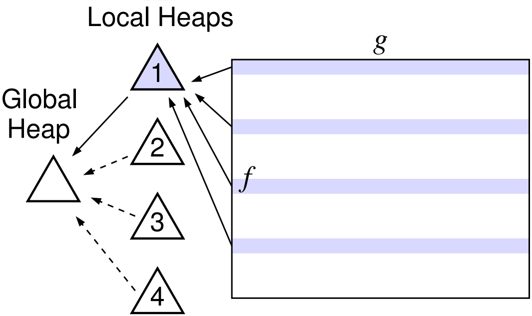
Intel Core i7, quad core, shared memory.

Parallel Multiplication Algorithm



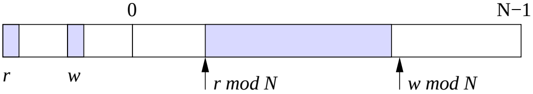
One thread per core.
Add results
in global heap.

Parallel Multiplication Algorithm



One thread per core. Add results in global heap.

Threads write to a finite circular buffer.



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

Old multiplication and factorization benchmark.

Intel Core i5 750 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	0.053	0.029	0.30	0.58	4.79
$p_3 := f_3(f_3+1)$	26.76	0.422	0.167	4.09	6.96	50.36
$p_4 := f_4(f_4+1)$	95.97	1.810	0.632	13.25	30.64	273.01
factor	Hensel lifting is mostly polynomial multiplication!!					
p_1 12341 terms	31.10	2.66	2.54	6.15	12.28	11.82
p_3 38711 terms	391.44	15.70	13.47	117.53	97.10	164.50
p_4 135751 terms	2953.54	56.68	44.06	332.86	404.86	655.49

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

$f_3 = (1+x+y+z)^{30}+1$ 5456 terms

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

Why is parallel speedup so poor?

To expand sums $f \times g$ Maple calls `` `expand/bigprod(f,g)` `` if

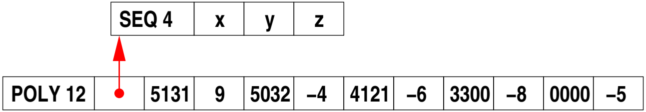
$$\#f > 2 \text{ and } \#g > 2 \text{ and } and\ \#f \times \#g > 1500.$$

```
`expand/bigprod` := proc(a,b)  # multiply two large sums
   if type(a,polynom(integer)) and type(b,polynom(integer)) then
     x := [op(indets(a) union indets(b))];
     d := max(op(map2(degree, a, x) + map2(degree, b, x)));
     k := iquo(kernelopts(wordsize), ilog2(d)+1 ); # bits per field
     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
   ...
```

sdmp:-Export $\implies$ `simpl(C)` $\implies$ shellsort, etc.

POLY the default representation in Maple.

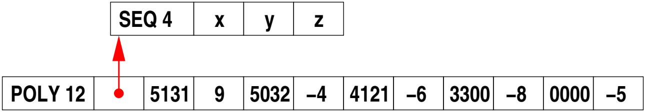
If all monomials pack into one word use



otherwise use the sum-of-products structure.

POLY the default representation in Maple.

If all monomials pack into one word use



otherwise use the sum-of-products structure.

But must reprogram entire kernel for new POLY !

$O(n)$	`f; degree(f); has(f,z); indets(f);`
$O(t)$	`degree(f,x); diff(f,x); expand(x*t);`

For f with t terms in n variables and $t \geq n$.

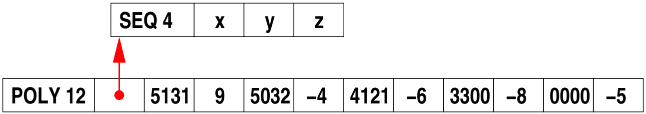
We use American flag sort, an in-place radix sort.

Everything except op and map is fast.

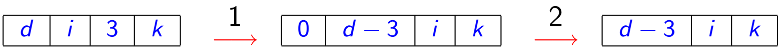
command	Maple 16	Maple 17	speedup	notes
coeff$(f,x,20)$	2.140 s	0.005 s	420x	terms easy to locate
coeffs(f,x)	0.979 s	0.119 s	8x	reorder exponents and radix
frontend$(g,[f])$	3.730 s	0.000 s	$\to O(n)$	looks at variables only
degree(f,x)	0.073 s	0.003 s	24x	stop early using monomial de
diff(f,x)	0.956 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	$\to O(n)$	first word in dag
op(f)	0.634 s	1.740 s	0.36x	converts to sum-of-products
simpl(f)	0.898 s	0.009 s	100x	only one object - already sor
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	$\to O(n)$	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])):
```



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



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)

New 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 2.16-8	Singular 3.1.4
$p_1 := f_1(f_1+1)$	0.053	0.029	0.042	0.017	0.30	0.57
$p_3 := f_3(f_3+1)$	0.422	0.167	0.398	0.137	4.09	6.77
$p_4 := f_4(f_4+1)$	1.810	0.632	1.730	0.508	13.25	30.99
factor			**Singular's factorization improved!**			
p_1 12341 terms	2.66	2.54	1.06	0.93	6.15	2.01
p_3 38711 terms	15.70	13.47	8.22	6.13	117.53	12.48
p_4 135751 terms	56.68	44.06	26.43	16.17	332.86	61.85

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

$f_3 = (1+x+y+z)^{30}+1$ 5456 terms

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

Profile for `factor(p1);` for 1 core.

function	Maple 16 time	Maple 16 time%	New Maple time	New Maple time%	Faster coeftayl time	Faster coeftayl time%
coeftayl	1.086s	41.06	0.310s	28.21	0.095s	12.03
expand	0.506s	19.13	0.263s	23.93	0.255s	32.28
diophant	0.424s	16.03	0.403s	34.94	0.299s	37.85
divide	0.256s	9.68	0.034s	3.09	0.035s	4.43
factor	0.201s	7.60	0.011s	1.00	0.010s	1.27
factor/hensel	0.127s	4.80	0.064s	5.82	0.063s	7.97
factor/unifactor	0.045s	1.70	0.033s	3.00	0.033s	4.18
total:	2.645s	100.00%	1.099s	100.00%	0.790s	100.00%

`coeftayl(f,x=a,k);` computes the coefficient of $(x-a)^k$ in f using `eval(diff(f,x$k),x=a)/k!` which is 3.5x faster.

But `add(coeff(f,x,i)` a^i `binomial(i,k), i=1..degree(f,x))` is 3x faster again!

Latest timings for factorization benchmark.

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

factor	Maple 16 1 core	Maple 16 4 cores	Maple 17 1 core (best)	Maple 17 4 cores (best)	Singular 3.1.4
p_1 12341 terms	2.66	2.54	1.06 (0.75)	0.94 (0.62)	2.01
p_3 38711 terms	15.70	13.47	8.22 (6.46)	6.13 (4.32)	12.48
p_4 135751 terms	56.68	44.06	26.43 (23.20)	16.17 (12.94)	61.85

With improvements to coeftayl and factor/diophant.

Reflecting on the gain?

1 core: $56.68 - 23.20 = 33.48$ and $\frac{56.68}{23.20} = 2.44x$

4 cores: $44.06 - 12.94 = 31.12$ and $\frac{44.06}{12.94} = 3.40x$.

Notes on the new integration for Maple 17.

We store f using POLY if

(i) f is an expanded polynomial, in names, with integer coefficients

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

(iii) 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 old sum-of-products representation.

We store f using POLY if

(i) f is an expanded polynomial, in names, with integer coefficients

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

(iii) 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 old sum-of-products representation.

- Packing is fixed by $n = \#\text{variables}$.

Notes on the new integration for Maple 17.

We store f using POLY if

(i) f is an expanded polynomial, in names, with integer coefficients

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

(iii) 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 old sum-of-products representation.

- Packing is fixed by $n = \#\text{variables}$.
- If $n = 8$, (iii) $\implies$ we use $b = \lfloor 64/9 \rfloor = 7$ bits per exponent field hence POLY restricts $d < 128$.

Notes on the new integration for Maple 17.

We store f using POLY if

(i) f is an expanded polynomial, in names, with integer coefficients

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

(iii) 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 old sum-of-products representation.

- Packing is fixed by $n = \#\text{variables}$.
- If $n = 8$, (iii) $\implies$ we use $b = \lfloor 64/9 \rfloor = 7$ bits per exponent field hence POLY restricts $d < 128$.
- The representation is invisible to the Maple user.
Conversions are automatic.

POLY polynomials are displayed in sorted order.

```
> f := 1+x+y;
```

$$f := 1 + x + y$$

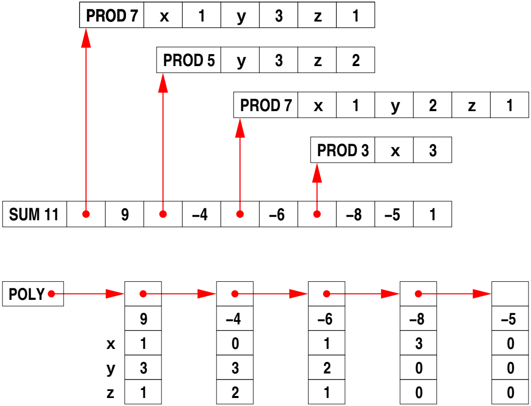
```
> g := 1-y*x+y^3;
```

$$g := y^3 - xy + 1$$

```
> dismantle(g);
POLY(8)
   EXPSEQ(3)
      NAME(4): x
      NAME(4): y
   DEGREES(HW): ^3 ^0 ^3
   INTPOS(2): 1
   DEGREES(HW): ^2 ^1 ^1
   INTNEG(2): -1
   DEGREES(HW): ^0 ^0 ^0
   INTPOS(2): 1
```

Reflections

We will not get high performance using these



Reflections

Amdahl's law: a harsh mistriss

$$\text{speedup} \leq \frac{1}{S + (1 - S)/N}$$

$N = \#cores$

$S = \text{overhead}\%$

Reflections

Amdahl's law: a harsh mistriss

$$\text{speedup} \le \frac{1}{S + (1-S)/N}$$

$N = \#cores$
$S = \text{overhead}\%$

overhead	50%	$\rightarrow$	5%	$\rightarrow$	1%
speedup ($N = 4$)	1.6x	$\rightarrow$	3.5x	$\rightarrow$	3.9x
speedup ($N = 16$)	1.9x	$\rightarrow$	9.1x	$\rightarrow$	13.9x

Reflections

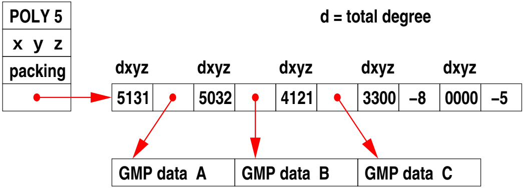
Amdahl's law:
a harsh mistriss

$$\text{speedup} \le \frac{1}{S + (1-S)/N}$$

$N = \#cores$
$S = $ overhead%

overhead	50%	→	5%	→	1%
speedup ($N = 4$)	1.6x	→	3.5x	→	3.9x
speedup ($N = 16$)	1.9x	→	9.1x	→	13.9x

Improve **simpl**!
$O(t^{3/4})$ hashalg
American flag sort



Future Work

What about these?

$$x^2 + \frac{2}{3}\,x - \frac{17}{9} \quad \text{and} \quad y^2 - 2.31\,y + 1.29$$

$$x^4 - t\,RootOf(\_Z^2 - t)\,x^2 + 3t \quad \text{and} \quad y''(x) - c\,y'(x) + 3$$

$$1 + x_1^8 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10}$$
$$+ x_{11}x_{12} + x_{13}x_{14} + x_{15}x_{16} + x_{17}x_{18} + x_{19}x_{20}$$