Sparse Polynomials in Maple

Roman Pearce

CECM/SFU

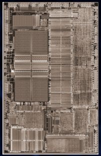
Joint work with Michael Monagan, Simon Fraser University.

Supported by MITACS, NSERC, and Maplesoft.

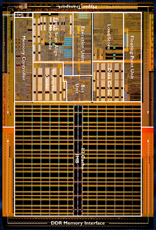
Moore's Law Has Not Stopped (Yet?)

- Most software is not getting faster
- *Throughput is still increasing!*
- Slow software is at risk!

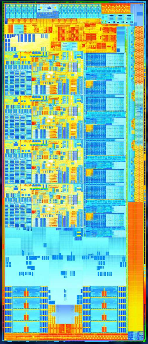
1 core
@ 1 GHz



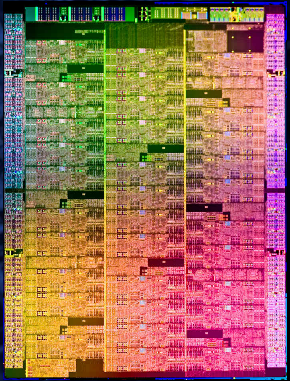
2 cores
@ 2 GHz



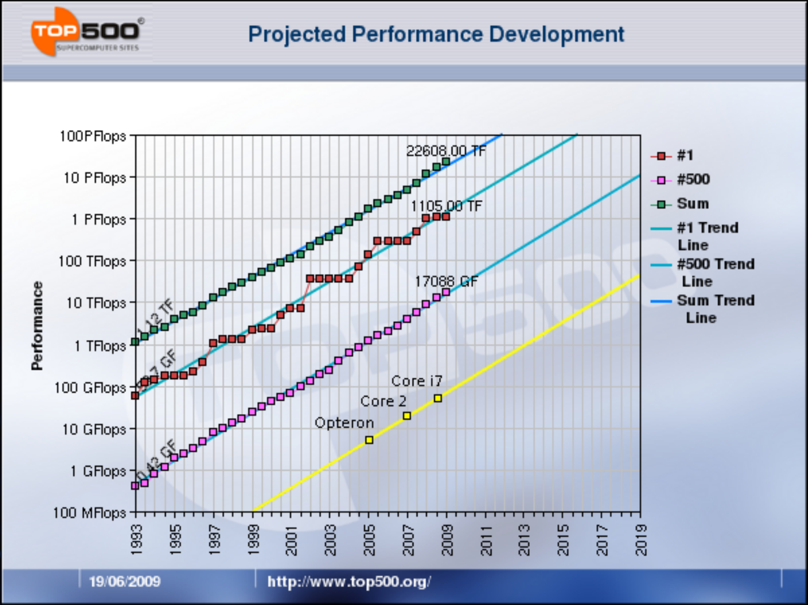
4 cores
@ 3 GHz



more cores, less GHz



High End Workstation ~ 16 Year Old Supercomputer



Two Grand Challenges

Scalability of *algorithms*

- high performance / parallel / GPU programming

Scalability of *systems*

- concurrency / overhead / Amdahl's Law

$$speedup \leq \frac{1}{S + (1 - S)/N} \qquad N = \#\text{ cores} \quad S = \text{overhead}$$

If we speed up a task by a factor of $N = 100$:

overhead	20%	10%	4%	2%	1%	0.5%	0.1%
speedup	4.8x	9x	20x	33x	50x	67x	91x

Maple's Strategy

Maple is:

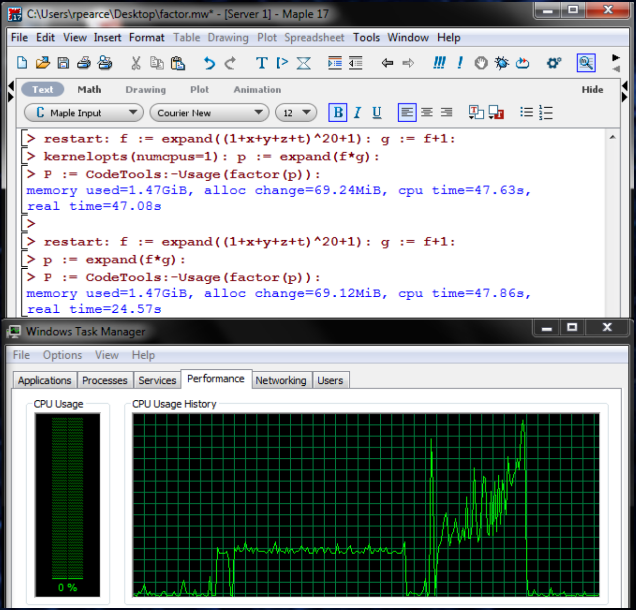
- compact kernel with C libraries
- 95% high level library code
- phones to workstations

First, build the platform:

- focus on high performance, parallel C
- cpu and memory *must not increase*
- redevelop higher level algorithms

Efficiency and interactive response are key.

Maple 17 on a Phenom II X4 Laptop



Benchmarks

	Maple 13 1 core	Maple 16 1 core	Maple 16 4 cores	Maple 17 1 core	Maple 17 4 cores	Magma 2.19-1	Singular 3-1-6
multiply							
$p_1 := f_1 \cdot g_1$	1.561	0.063	0.030	0.041	0.012	0.330	0.585
$p_4 := f_4 \cdot g_4$	98.351	2.180	0.649	1.814	0.416	13.700	31.806
$p_5 := f_5 \cdot g_5$	13.666	1.588	0.384	0.153	0.154	13.240	17.776
$p_6 := f_6 \cdot g_6$	11.486	0.772	0.628	0.204	0.082	0.890	1.787
divide							
$q_1 := p_1/f_1$	1.451	0.065	0.033	0.042	0.015	0.360	0.183
$q_4 := p_4/f_4$	92.867	2.253	0.736	1.842	0.483	18.540	11.420
$q_5 := p_5/f_5$	5.570	1.636	0.417	1.445	0.333	12.480	10.478
$q_6 := p_6/f_6$	10.421	0.769	0.627	0.215	0.095	7.900	1.484
factor							
p_1 (12341)	31.330	2.792	2.658	0.790	0.650	6.510	0.853
p_4 (135751)	2856.388	59.009	46.151	24.345	12.733	320.040	39.353
p_5 (12552)	302.453	26.435	16.152	12.131	6.800	105.550	9.604
p_6 (417311)	1359.473	51.702	48.808	8.295	6.330	369.120	20.603

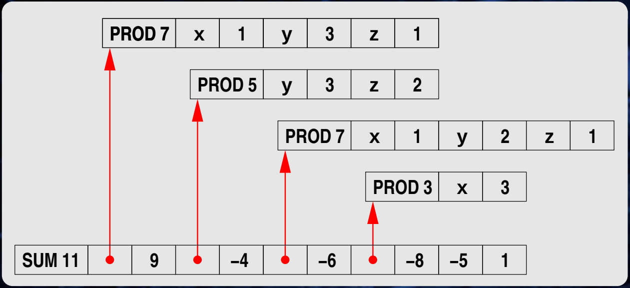
$$f_1 = (1+x+y+z)^{20}+1$$
$$g_1 = (1+x+y+z)^{20}+2$$
1771 terms, small

$$f_4 = (1+x+y+z+t)^{20}+1$$
$$g_4 = (1+x+y+z+t)^{20}+2$$
10626 terms, large

$$f_5 = (1+x)^{20}(1+y)^{20}(1+z)^{20}+1$$
$$g_5 = (1-x)^{20}(1-y)^{20}(1-z)^{20}+1$$
9261 terms, dense

$$f_6 = (1+u^2+v+w^2+x-y)^{10}+1$$
$$g_6 = (1+u+v^2+w+x^2-y)^{10}+1$$
3003 terms, sparse

Maple 16 Overhead

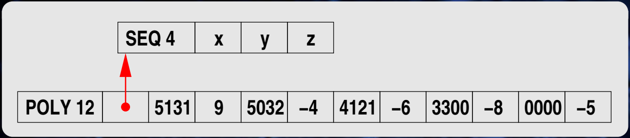


$$f = 9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$$

Maple was designed in a different era:

- code size restrictions
- small, complex objects
- short recursive programs

Maple 17 Data Structure



$$f = 9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$$

- used by default for $\mathbb{Z}[x_1, x_2, \ldots, x_n]$
- terms are sorted in graded lex order
- total degree + exponents in one word

#vars	1	2	3	4	5	6	7	8	
degree	2^{62}	2^{21}	2^{16}	2^{12}	1024	512	256	128	...

Kernel Operations

$f := expand(mul(randpoly(i, degree = 100, dense), i = [x, y, z]))$: **$10^6$ terms**

command	description	SUM	POLY	**speedup**
f;	evaluation	0.210 s	0.000 s	$\rightarrow$ **O(v)**
$coeff(f, x, 20)$	coefficient of x^{20}	1.280 s	0.007 s	**182x**
$coeffs(f, x)$	all coefficients in x	0.716 s	0.069 s	**10x**
$degree(f, x)$	degree in x	0.159 s	0.008 s	**20x**
$degree(f)$	total degree	0.246 s	0.000 s	$\rightarrow$ **O(1)**
$diff(f, x)$	differentiate wrt x	0.778 s	0.023 s	**33x**
$expand(2xf)$	multiply by a term	0.897 s	0.044 s	**20x**
$has(f, x^{101})$	find subexpression	0.136 s	0.003 s	**45x**
$indets(f)$	set of indeterminates	0.164 s	0.000 s	$\rightarrow$ **O(1)**
$lcoeff(f, x)$	leading coefficient in x	0.149 s	0.007 s	**21x**
$subs(x = y, f)$	replace variable	0.864 s	0.048 s	**18x**
$type(f, polynom)$	type check	0.132 s	0.000 s	$\rightarrow$ **O(v)**

Sub-linear operations key vs. Amdahl's Law

Bit Level Programming (Hacker's Delight)

We pack $p = x^3 y^4 z^5$ in $[x, y, z]$ as $[12, 3, 4, 5]$.
Imagine we use 4 bits each: 1100 0011 0100 0101.

How to test if p is divisible by $q = x^4 y^3 z^2 = [9, 4, 3, 2]$.
$(p - q)$ **XOR** p **XOR** $q =$ **bits borrowed.**

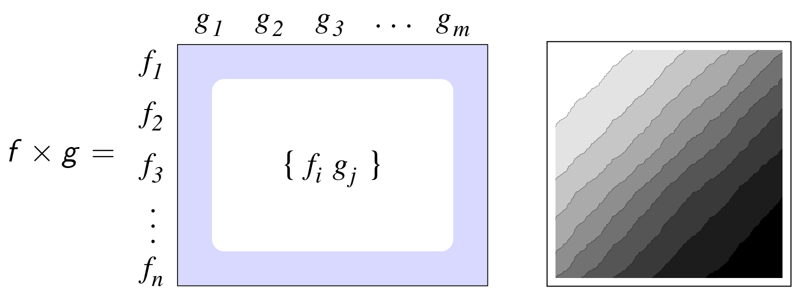
```
  1100 0011 0100 0101 = p
- 1001 0100 0011 0010 = q
---------------------
  0010 1111 0001 0011 = p-q

  0111 1000 0110 0100 = (p-q) XOR p XOR q
  0001 0001 0001 0000 = underflow mask
---------------------
  0001 0000 0000 0000 = underflow in x
```

Fast test for monomial division in a subset of variables.

Sparse Polynomial Multiplication

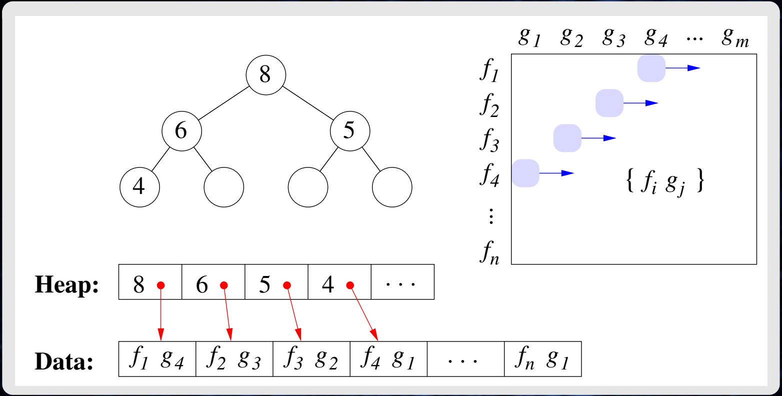
Merge all products $f_i \cdot g_j$



- cache locality $\iff$ order of products
- exploit monomial ordering:

$$f_i \cdot g_j > f_i \cdot g_{j+1} \quad \text{and} \quad f_i \cdot g_j > f_{i+1} \cdot g_j$$

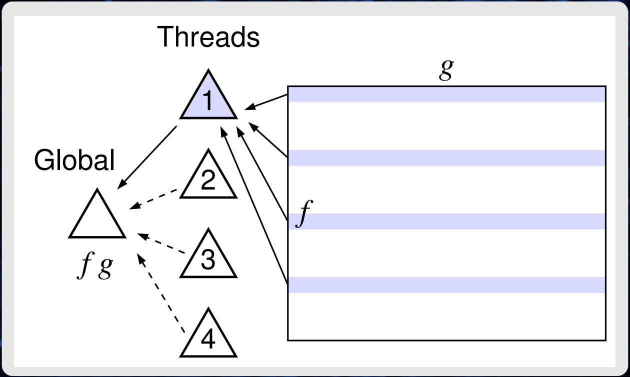
... Using a Heap



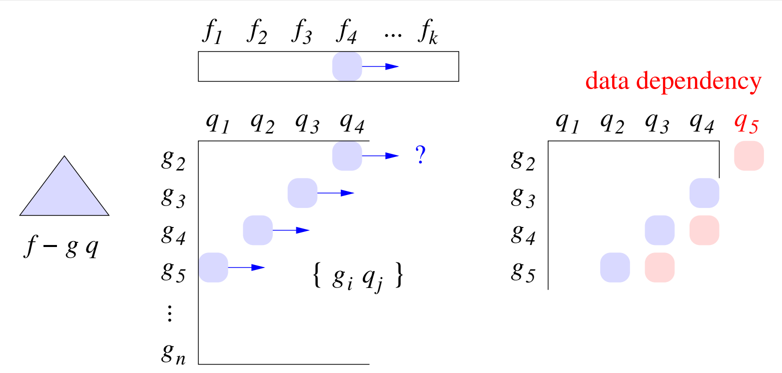
- small working set: $O(\#f) + 1$
  division $f - qg = r$ uses $O(\#g + \#q + \#r)$ memory
- combine like elements in the heap
- *organize data by frequency of access!*

Parallel Multiplication

- divide up the working set $\rightarrow$ superlinear speedup
- threads write to buffers to avoid memory blowup
- use master work to cooperatively balance the load

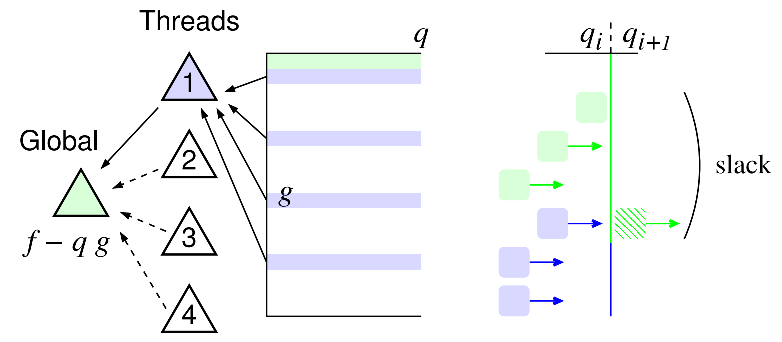


Data Dependencies in Division



- merge $g_2 q_4$ but can't insert $g_2 q_5$
- q_5 does not exist yet, it is computed later
- threads must determine when it is safe to proceed
- data dependency $\Longrightarrow$ communication bottleneck

Parallel Division



- move tight dependencies into master task
  $\implies$ dependencies serialize the algorithm
- cover up communication latency with slack
- master task can steal extra work

Sparse Powering (Tom Coates)

$$f = xy^3z^2 + x^2y^2z + xy^3z + xy^2z^2 + y^3z^2 + y^3z + 2y^2z^2 + 2xyz + y^2z + yz^2 + y^2 + 2yz + z$$

Expand f^{50} (472226 terms out of a possible 1.54 million

... also $f^{100}, f^{200}, f^{500}, \ldots$ *and with more variables!*

Algorithm: $f \cdot f \cdot f \cdots f$ *?!*

k	terms f^k	Maple 16	Magma 2.17	Singular 3.1.4	our multiply
10	4246	0.030	0.010	0.010	0.000
20	31591	0.403	0.210	0.240	0.030
30	104036	2.537	1.200	1.470	0.260
40	243581	9.062	3.620	4.930	0.970
50	472226	23.131	9.260	12.460	2.620
60	811971	49.572	19.100	26.660	5.730
70	1284816	95.654	36.390	50.180	10.950
250	57636126	–	–	–	40 min

Why Multiplying is Slow

$$f = xy^3z^2 + x^2y^2z + xy^3z + xy^2z^2 + y^3z^2 + y^3z + 2y^2z^2 + 2xyz + y^2z + yz^2 + y^2 + 2yz + z$$

It slowly builds the result (number of terms)

i	$f^{i-1} \times f$	=	f^i
2	13 x 13	=	58
3	58 x 13	=	158
4	158 x 13	=	335
5	335 x 13	=	611
	...		
10	3145 x 13	=	4246
11	4246 x 13	=	5578
12	5578 x 13	=	7163
13	7163 x 13	=	9023

i	$f^{i-1} \times f$	=	f^i
20	27190 x 13	=	31591
21	31591 x 13	=	36443
22	36443 x 13	=	41768
23	41768 x 13	=	47588
	...		
40	225980 x 13	=	243581
41	243581 x 13	=	262073
42	262073 x 13	=	281478
43	281478 x 13	=	301818

Square and Multiply is Worse

i	$f^{i/2} \times f^{i/2}$	=	f^i	time
2	13 x 13	=	58	0.000
4	58 x 58	=	335	0.000
8	335 x 335	=	2253	0.000
16	2253 x 2253	=	16473	0.090
32	16473 x 16473	=	125873	5.460
64	125873 x 125873	=	983905	19 min

Dense arithmetic? $x \to t,\ y \to t^{2k+1},\ z \to t^{3(2k+1)k+1}$

k	$\deg(f, t)$	Magma 2.17	sparse mul	new method
40	19686	1.470	0.968	0.159
70	59646	28.260	10.833	0.941
100	121206	93.640	48.932	3.026
150	271806	–	276.320	10.880
250	753006	–	40 min	68.626

Main Result on Powering

We compute $\mathbf{f^k}$ for $\sim$ cost of $\mathbf{f^{k-1}} \times \mathbf{f}$.

Idea from Euler's formula for power series:

$$f = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$$

$$g_0 = f_0^k$$

$$g_i = \frac{1}{i f_0} \sum_{j=1}^{\min(d,i)} ((k+1)j - i) f_j g_{i-j} \quad \text{for} \quad i = 1 \ldots kd \quad \in \mathbf{O(kd^2)}$$

$g = f^3$		1	$9x$	$33x^2$	$63x^3$	$66x^4$	$36x^5$	$8x^6$
	1							
f	$3x$		$9x$	$54x^2$	$99x^3$	$0x^4$	$-198x^5$	$-216x^6$
	$2x^2$		$12x^2$	$90x^3$	$264x^4$	$378x^5$	$264x^6$	

□

Main Result on Powering

We compute $\mathbf{f^k}$ for $\sim$ cost of $\mathbf{f^{k-1}} \times \mathbf{f}$.

Idea from Euler's formula for power series:

$$f = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$$

$$g_0 = f_0^k$$

$$g_i = \frac{1}{i f_0} \sum_{j=1}^{\min(d,i)} ((k+1)j - i) f_j g_{i-j} \quad \text{for} \quad i = 1 \ldots kd \quad \in \mathbf{O(kd^2)}$$

	$g = f^3$	1	$9x$	$33x^2$	$63x^3$	$66x^4$	$36x^5$	$8x^6$
	1							
f	$3x$	$9x$	$54x^2$	$99x^3$	$0x^4$	$-198x^5$	$-216x^6$	
	$2x^2$	$12x^2$	$90x^3$	$264x^4$	$378x^5$	$264x^6$		□

Main Result on Powering

We compute $\mathbf{f^k}$ for $\sim$ cost of $\mathbf{f^{k-1}} \times \mathbf{f}$.

Idea from Euler's formula for power series:

$$f = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$$

$$g_0 = f_0^k$$

$$g_i = \frac{1}{i f_0} \sum_{j=1}^{\min(d,i)} ((k+1)j - i) f_j g_{i-j} \quad \text{for} \quad i = 1 \ldots kd \quad \in \mathbf{O(kd^2)}$$

	$g = f^3$	1	$9x$	$33x^2$	$63x^3$	$66x^4$	$36x^5$	$8x^6$
	1							
f	$3x$	$9x$	$54x^2$	$99x^3$	$0x^4$	$-198x^5$	$-216x^6$	
	$2x^2$	$12x^2$	$90x^3$	$264x^4$	$378x^5$	$264x^6$		□

Main Result on Powering

We compute $\mathbf{f^k}$ for $\sim$ cost of $\mathbf{f^{k-1}} \times \mathbf{f}$.

Idea from Euler's formula for power series:

$$f = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$$

$$g_0 = f_0^k$$

$$g_i = \frac{1}{i f_0} \sum_{j=1}^{\min(d,i)} ((k+1)j - i) f_j g_{i-j} \quad \text{for} \quad i = 1 \ldots kd \quad \in \mathbf{O(kd^2)}$$

	$g = f^3$	1	$9x$	$33x^2$	$63x^3$	$66x^4$	$36x^5$	$8x^6$
f	1							
	$3x$	$9x$	$54x^2$	$99x^3$	$0x^4$	$-198x^5$	$-216x^6$	
	$2x^2$	$12x^2$	$90x^3$	$264x^4$	$378x^5$	$264x^6$		□

Main Result on Powering

We compute $\mathbf{f^k}$ for $\sim$ cost of $\mathbf{f^{k-1}} \times \mathbf{f}$.

Idea from Euler's formula for power series:

$$f = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$$

$$g_0 = f_0^k$$

$$g_i = \frac{1}{i f_0} \sum_{j=1}^{\min(d,i)} ((k+1)j - i) f_j g_{i-j} \quad \text{for} \quad i = 1 \ldots kd \quad \in \mathbf{O(kd^2)}$$

	$g = f^3$	1	$9x$	$33x^2$	$63x^3$	$66x^4$	$36x^5$	$8x^6$
	1							
f	$3x$	$9x$	$54x^2$	$99x^3$	$0x^4$	$-198x^5$	$-216x^6$	
	$2x^2$	$12x^2$	$90x^3$	$264x^4$	$378x^5$	$264x^6$		□

Main Result on Powering

We compute $\mathbf{f^k}$ for $\sim$ cost of $\mathbf{f^{k-1}} \times \mathbf{f}$.

Idea from Euler's formula for power series:

$$f = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$$

$$g_0 = f_0^k$$

$$g_i = \frac{1}{i f_0} \sum_{j=1}^{\min(d,i)} ((k+1)j - i) f_j g_{i-j} \quad \text{for} \quad i = 1 \ldots kd \quad \in \mathbf{O(kd^2)}$$

$g = f^3$		1	$9x$	$33x^2$	$63x^3$	$66x^4$	$36x^5$	$8x^6$
	1							
f	$3x$	$9x$	$54x^2$	$99x^3$	$0x^4$	$-198x^5$	$-216x^6$	
	$2x^2$	$12x^2$	$90x^3$	$264x^4$	$378x^5$	$264x^6$		

Main Result on Powering

We compute $\mathbf{f^k}$ for $\sim$ cost of $\mathbf{f^{k-1}} \times \mathbf{f}$.

Idea from Euler's formula for power series:

$$f = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$$

$$g_0 = f_0^k$$

$$g_i = \frac{1}{i f_0} \sum_{j=1}^{\min(d,i)} ((k+1)j - i) f_j g_{i-j} \quad \text{for } \; i = 1 \ldots kd \;\; \in \mathbf{O(kd^2)}$$

	$g = f^3$	1	$9x$	$33x^2$	$63x^3$	$66x^4$	$36x^5$	$8x^6$
	1							
f	$3x$	$9x$	$54x^2$	$99x^3$	$0x^4$	$-198x^5$	$-216x^6$	
	$2x^2$	$12x^2$	$90x^3$	$264x^4$	$378x^5$	$264x^6$		□

Main Result on Powering

We compute $\mathbf{f^k}$ for $\sim$ cost of $\mathbf{f^{k-1}} \times \mathbf{f}$.

Idea from Euler's formula for power series:

$$f = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$$

$$g_0 = f_0^k$$

$$g_i = \frac{1}{i f_0} \sum_{j=1}^{\min(d,i)} ((k+1)j - i) f_j g_{i-j} \quad \text{for} \quad i = 1 \ldots kd \quad \in \mathbf{O(kd^2)}$$

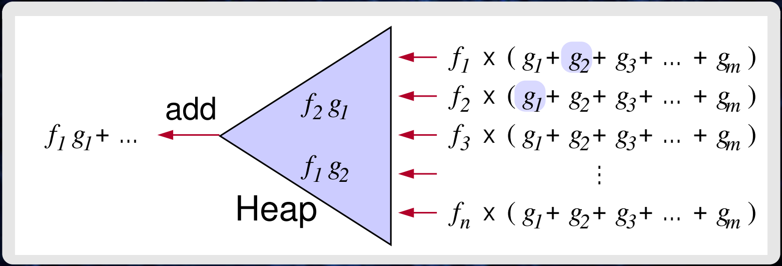
	$g = f^3$	1	$9x$	$33x^2$	$63x^3$	$66x^4$	$36x^5$	$8x^6$
	1							
f	$3x$	$9x$	$54x^2$	$99x^3$	$0x^4$	$-198x^5$	$-216x^6$	
	$2x^2$	$12x^2$	$90x^3$	$264x^4$	$378x^5$	$264x^6$		

Sparse Powering

For multivariate polynomials we use Kronecker substitution.

Merge only products where f_i and g_{i-j} non-zero:

$$g_i = \frac{1}{if_0} \sum_{j=1}^{\min(d,i)} ((k+1)j - i) f_j g_{i-j} \quad \text{for} \quad i = 1 \ldots kd.$$



Problems: $\mathbb{Z}_p$, and redundant products whose sum is zero.

Key Improvement

Euler's method multiplies

$$(\text{terms of } f) \times (\text{terms of } f^{k-1})$$

to compute $\mathbf{f^{k-1}}$.

But it can output $\mathbf{f^{k}}$ almost for free!

$\Longrightarrow$ FPS algorithm

Key Improvement

Euler's method multiplies

$$(\text{terms of } f) \times (\text{terms of } f^{k-1})$$

to compute $\mathbf{f^{k-1}}$.

But it can output $\mathbf{f^{k}}$ almost for free!

$\Longrightarrow$ FPS algorithm

$f = c_1x_1 + c_2x_2 + \cdots + c_tx_t$. f^k generates $\binom{t+k-1}{k}$ terms.

t	k	Magma	Singular	multiply	binomial	**FPS**
3	100	0.010	0.050	0.026	0.001	0.001
3	500	3.480	12.750	4.560	0.055	0.069
4	50	0.120	0.180	0.033	0.005	0.007
4	200	74.360	44.610	13.151	0.521	0.714
6	30	63.270	1.170	0.173	0.039	0.057
6	40	–	6.670	1.471	0.222	0.531
8	25	–	10.700	1.504	0.452	0.649
8	35	–	148.970	28.342	5.927	13.828

Future Work

- Parallelize Maple!
- Haswell BMI2 instructions
- dense methods: GPU and OpenCL
- sparse methods: normal form, eval, interp, ...
- new areas: sparse linear algebra, Gröbner bases

Discuss: Ideas for Magma