Workpackage 5:
High Performance Mathematical Computing

Clément Pernet

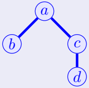
Final OpenDreamKit Project review

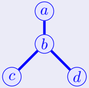
Luxembourg, October 30, 2019

High Performance Mathematical Computing

Mathematical computing

Computing with a large variety of objects

- $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q,$ $\quad 17541718814389012164632$
- Polynomials over $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q,$ $\quad \frac{2}{5}x^3 + x^2 - \frac{1}{19}x + 2$
- Matrices over $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q,$ $\quad \begin{bmatrix} 27 & 3 & -1 \\ 9 & 0 & 2 \end{bmatrix}$
- Matrices of polynomials over $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q,$ $\quad \begin{bmatrix} 3x^2+3 & 2x^2+3 \\ 4x^2+1 & x^2+4x+4 \end{bmatrix}$
- Tree algebras $\quad \frac{3q^2-q^5}{q^5+2q^4+3q^3+3q^2+2q+1}$ 

$+ \frac{2q}{q^4+q^3+2q^2+q+1}$ 

for applications where all digits matter (most often).

High performance mathematical computing

Need for High performance: applications where size is crucial:

Experimental maths: testing conjectures

- larger instances give higher confidence

High performance mathematical computing

Need for High performance: applications where size is crucial:

Experimental maths: testing conjectures

- larger instances give higher confidence

Algebraic cryptanalysis: security = computational difficulty

- key size determined by the largest solvable problem

Example

Breaking RSA by integer factorization: $n = pq$. Last record:

- n of 768 bits
- linear algebra in dimension $192\,796\,550$ over $\mathbb{F}_2$ (105Gb)
- About 2000 CPU years

High performance mathematical computing

Need for High performance: applications where size is crucial:

Experimental maths: testing conjectures

- larger instances give higher confidence

Algebraic cryptanalysis: security $=$ computational difficulty

- key size determined by the largest solvable problem

Example

Breaking RSA by integer factorization: $n = pq$. Last record:

- n of 768 bits
- linear algebra in dimension $192\,796\,550$ over $\mathbb{F}_2$ (105Gb)
- About 2000 CPU years

3D data analysis, shape recognition:

- via persistent homology
- large sparse matrices over $\mathbb{F}_2$, $\mathbb{Z}$

Goal: delivering high performance to maths users

Systems : GAP, PARI/GP, SageMath, Singular

Components : FLINT, MPIR, LinBox, PPL, NumPy

Goal: delivering high performance to maths users

Harnessing modern hardware ⇝ parallelisation

- in-core parallelism (SIMD vectorisation)
- multi-core parallelism
- distributed computing: clusters, cloud

Systems : GAP, PARI/GP, SageMath, Singular

Components : FLINT, MPIR, LinBox, PPL, NumPy

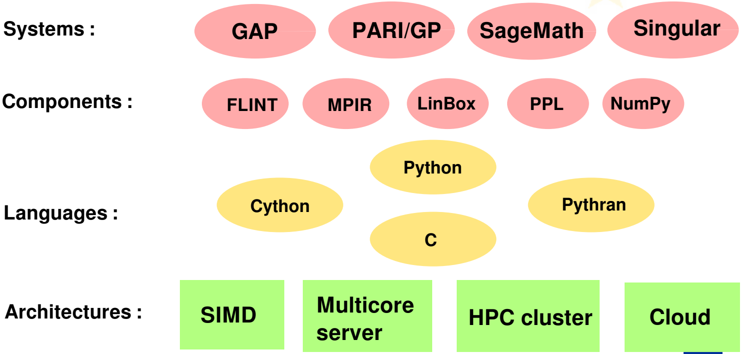
Architectures : SIMD, Multicore server, HPC cluster, Cloud

Goal: delivering high performance to maths users

Languages

- Computational Maths software uses high level languages (e.g. Python)
- High performance delivered by languages close to the metal (C, assembly)

$\leadsto$ compilation, automated optimisation



High performance mathematical computing

Goal:

- Improve/Develop parallelization of software components
- Expose them through the software stack
- Offer High Performance Computing to VRE's users

Computational Kernels

Linear algebra

Arithmetic

Computational Kernels

Linear algebra

Similarities with numerical HPC

- building block to which others reduce to
- (rather) simple algorithmic
- high compute/memory intensity

Specificities

- Multiprecision arithmetic → lifting from finite precision ($\mathbb{F}_p$)
- Rank deficiency → unbalanced dynamic blocking
- Early adopter of subcubic matrix arithemtic → recursion

Arithmetic

Computational Kernels

Linear algebra

Arithmetic

- Wide variety of computing domains: $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q, \mathbb{F}_q[X], \mathbb{Z}[X, Y, Z], ..$
- possibly with dynamic size

Challenge

- most often memory intensive operations → hard to parallelize
- very fine grain, but billions of instance → fine tuning

Outline

Workpackage management

T5.1: Number theory with PARI/GP

T5.2: Group theory with GAP

T5.3: Exact linear algebra with LinBox

T5.4: Polynomial arithmetic with Singular

Outline

Workpackage management

T5.1: Number theory with PARI/GP

T5.2: Group theory with GAP

T5.3: Exact linear algebra with LinBox

T5.4: Polynomial arithmetic with Singular

Outcome of WorkPackage 5

Component	Review 1	Review 2	Final review
T5.1 Pari/GP			D5.16
T5.2 GAP			D5.15
T5.3 LinBox		D5.12	D5.14
T5.4 Singular	D5.6, D5.7		D5.13
T5.5 MPIR	D5.5, D5.7		
T5.6 Combinatorics	D5.1	D5.11	
T5.7 Pythran	D5.2	D5.11	
T5.8 SunGrid Engine	D5.3		

Overall

- 31 software releases
- 16 research papers in journals or conference proceedings

Addressing recommendations of review 1 and 2

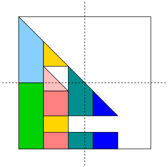
RP1 Recommendation 10: *Regarding WP5,* **make contacts** *with HPC community in order to ascertain current state-of-the-art. The work in this WP needs to be* **nearer the leading edge**.

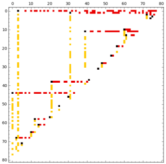
Addressing recommendations of review 1 and 2

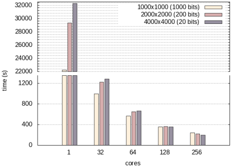
RP1 Recommendation 10: *Regarding WP5,* **make contacts** *with HPC community in order to ascertain current state-of-the-art. The work in this WP needs to be* **nearer the leading edge***.*

Leading edge achievements in linear algebra

- symmetric factorization outperforms LAPACK implementation
- new non-hierarchical generator for quasiseparable matrices
- large scale parallelization of rational linear solver







Addressing recommendations of review 1 and 2

RP2 Recommendation 1: *A minor aspect: In the deliverable D5.11, authors have to clarify the reason why the speedup with the use of cores is not so high when you increment the number of cores. The presentation has also to be improved.*

- D5.11 was complemented with a clarification, polished resubmitted after the review.

Addressing recommendations of review 1 and 2

RP2 Recommendation 1: *A minor aspect: In the deliverable D5.11, authors have to clarify the reason why the speedup with the use of cores is not so high when you increment the number of cores. The presentation has also to be improved.*

- D5.11 was complemented with a clarification, polished resubmitted after the review.

RP2 Recommendation 10: *Some guidelines (set of recommendations) for using the different hardware architectures would be recommendable.*

- A Blog post was produced as a use case and published on `opendreamkit.org`.

Outline

Workpackage management

T5.1: Number theory with PARI/GP

T5.2: Group theory with GAP

T5.3: Exact linear algebra with LinBox

T5.4: Polynomial arithmetic with Singular

PARI-GP

PARI ecosystem

PARI library: dedicated routines for number theory

PARI-GP: an interactive system

GP2C: a GP to C compiler

Generic parallelization engine for the whole suite

Delivers support for

- sequential computations
- POSIX threads
- MPI for distributed computing

Features:

- Same code base
- automated parallelization
- full control for power users/developpers

Main Achievements

- Fast linear algebra over the rationals and cyclotomic fields
- Fast Chinese remainders and multimodular reduction
- Parallel polynomial resultant
- Fast modular polynomials and applications
- MPQS integer factorization rewrite

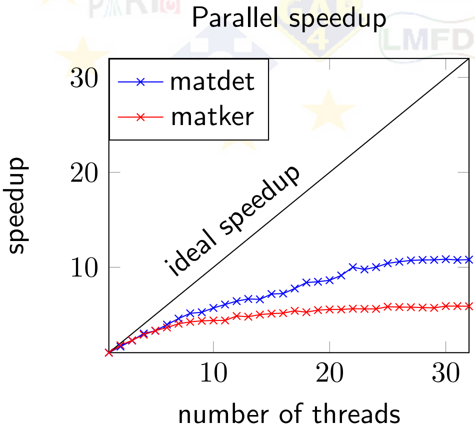
Well-honed strategy after preliminary assessment

- Creation of "worker" functions from existing code
- Insertion of actual parallel instructions
- Incremental buildup, independently instrumenting one high-level function at a time.

Highlight

Linear algebra over rationals:

- Required for cyclotomic rings
- Fast Chinese remaindering
- Fast CUP decomposition over finite fields
- Parallelization

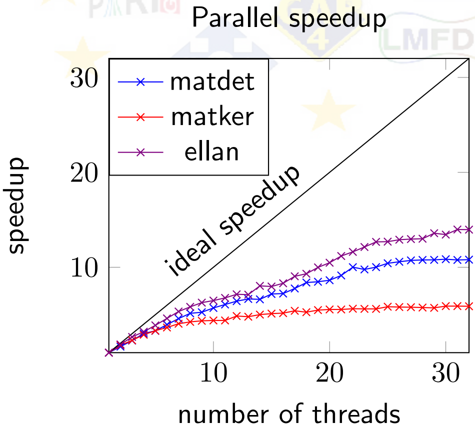


Highlight

Linear algebra over rationals:

- Required for cyclotomic rings
- Fast Chinese remaindering
- Fast CUP decomposition over finite fields
- Parallelization

Fourier transform of L-functions



Outline

Workpackage management

T5.1: Number theory with PARI/GP

T5.2: Group theory with GAP

T5.3: Exact linear algebra with LinBox

T5.4: Polynomial arithmetic with Singular

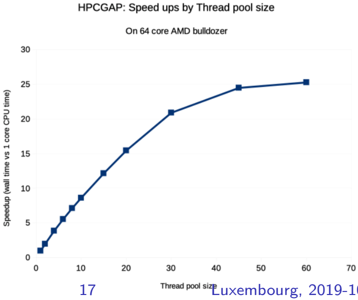
HPC GAP

HPC-GAP

- Multi-threaded GAP.
- Targets:
  - multicore servers.
  - good speedups
  - high level abstraction

Achievement

- Fork from GAP, diverged for a long time
- Huge effort to bring it back in:
  GAP 4.9.1 (Month 33).
  → first GAP release with HPCGAP integrated as a compile-time option



Dense linear algebra over small finite fields

- Matrix multiplication, Gaussian elimination, echelon forms, etc
- A key kernel for many GAP computations

MeatAxe64

A new C and assembler library, tuned for performance at all levels:

- new data representations and assembler kernels
- new algorithms for many fields
- control of cache usage and memory bandwidth
  → allowing for sharing between threads and cores
- purpose built highly efficient task farm
  → 1M x 1M dense matrix multiply over GF(2) in 8 hours (64 core AMD bulldozer).

Fully available from GAP.

GAP Demo

Outline

Workpackage management

T5.1: Number theory with PARI/GP

T5.2: Group theory with GAP

T5.3: Exact linear algebra with LinBox

T5.4: Polynomial arithmetic with Singular

Parallel Rational solver algorithmic

Method	Bit complexity
Gauss over $\mathbb{Q}$	$2^{O(n)}$
Gauss $\mod \text{bound}(sol)$	$O(n^5)$
Chinese Remaindering	$O\tilde{}(n^4)$
p-adic lifting	$O\tilde{}(n^3)$

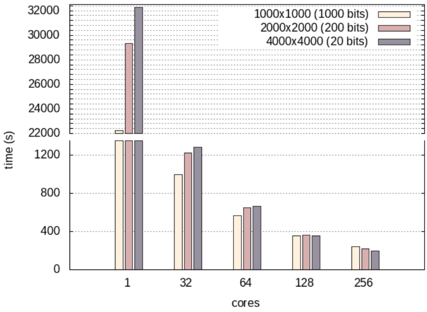
Chinese Remaindering

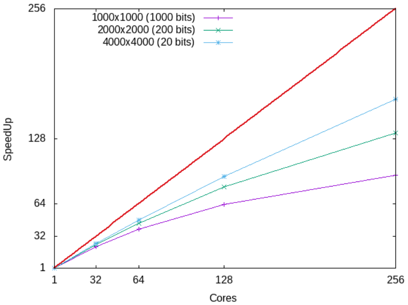
1. Solve the system independently modulo $p_1, p_2, \ldots, p_k$
2. Reconstruct a solution modulo $p_1 \times p_2 \times \ldots, p_k$.
3. Reconstruct over $\mathbb{Q}$

p-adic lifting

1. Solve the system modulo p
2. Iteratively lift the solution modulo $p^2, p^3, \ldots, p^k$
3. Reconstruct over $\mathbb{Q}$

Distributed memory Chinese Remaindering





Conclusions

- (almost) embarrassingly parallel
- but overwhelming computational cost ($O\tilde{}(n^4)$)
- hybrid OpenMP-MPI version slightly slower but better memory efficiency

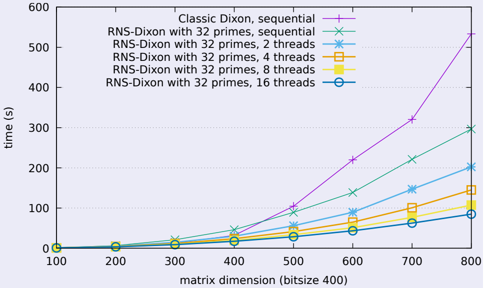
Shared memory p-adic lifting

A new hybrid algorithm: Chinese Remaindering within p-adic lifting

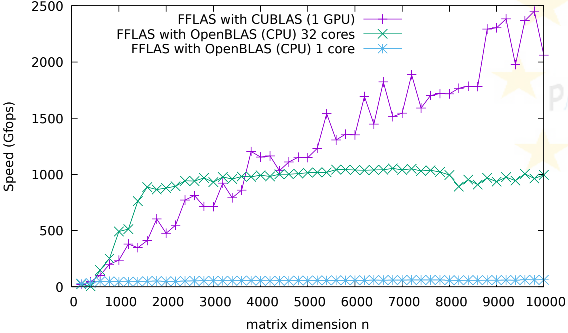
- Smaller critical path
- Higher degree of parallelism

Improving state of the art efficiency

- Improved sequential efficiency (memory access pattern, BLAS3)
- Chinese remaindering delivers good parallel scaling

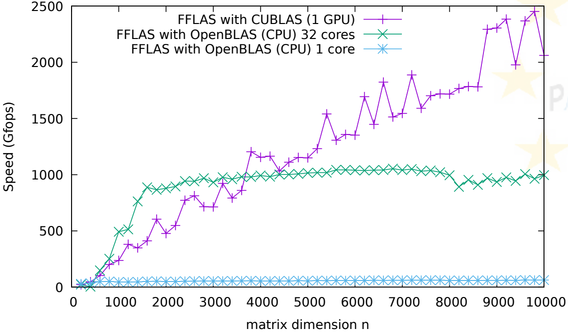


GPU enabled `fflas-ffpack`



- Matrix product over $\mathbb{Z}/131071\mathbb{Z}$
- 32 Skylake-X cores (AVX512)
- 1 Nvidia Tesla V100 GPU

GPU enabled `fflas-ffpack`



- Matrix product over $\mathbb{Z}/131071\mathbb{Z}$
- 32 Skylake-X cores (AVX512)
- 1 Nvidia Tesla V100 GPU

Limitations and perspectives

Bottleneck in the transfer between GPU and RAM

- deport more computations to the GPU → dedicated GPU kernels
- communication avoiding block scheduling → deep structural change

Outline

Workpackage management

T5.1: Number theory with PARI/GP

T5.2: Group theory with GAP

T5.3: Exact linear algebra with LinBox

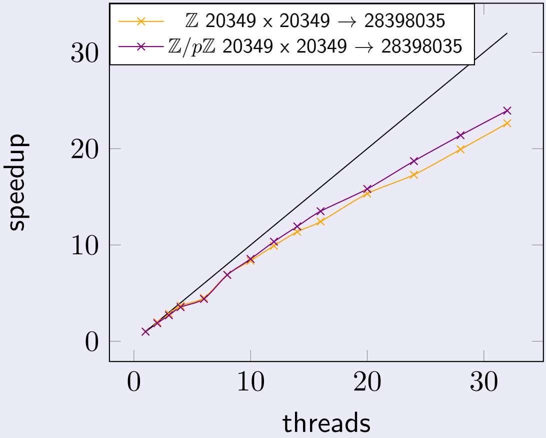
T5.4: Polynomial arithmetic with Singular

Singular demo

SINGULAR Demo

Main achievements

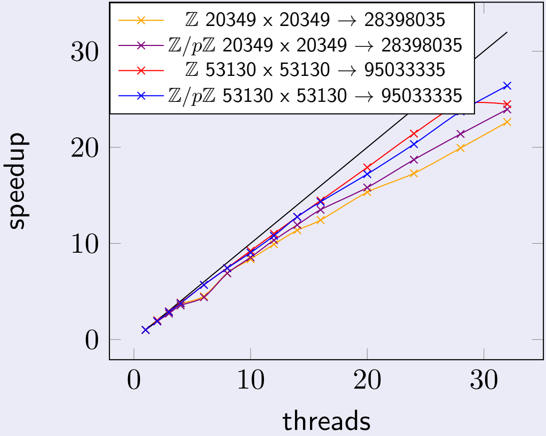
Multivariate Polynomial Multiplication in FLINT



- over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{F}_p$, $\mathbb{F}_{p^n}$ for $p < 2^{64}$.
- Lex, degLex and degGrevLex ordering supported
- Sequential alg: improves Singular's
- Close to linear scaling
- Singular now relies on FLINT

Main achievements

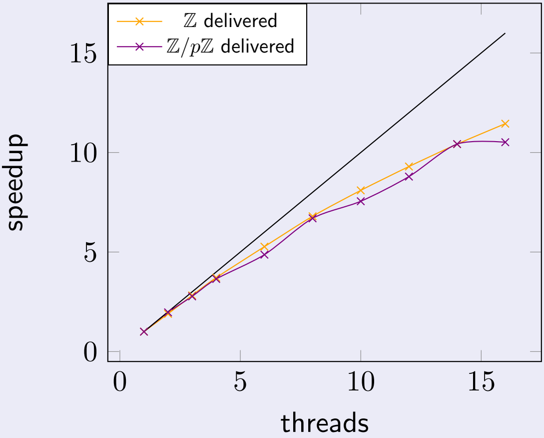
Multivariate Polynomial Multiplication in `FLINT`



- over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{F}_p$, $\mathbb{F}_{p^n}$ for $p < 2^{64}$.
- Lex, degLex and degGrevLex ordering supported
- Sequential alg: improves `Singular`'s
- Close to linear scaling
- `Singular` now relies on `FLINT`

Main achievements

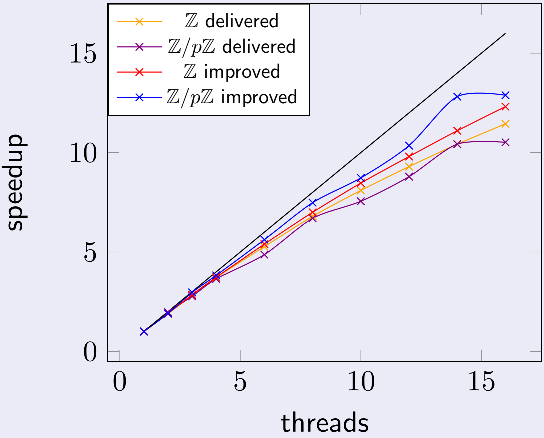
Multivariate Polynomial GCD in FLINT



- over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{F}_p$, $\mathbb{F}_{p^n}$ for $p < 2^{64}$.
- Lex, degLex and degGrevLex ordering supported
- Sequential alg: improves `Singular`'s
- Close to linear scaling
- `Singular` now relies on `FLINT`

Main achievements

Multivariate Polynomial GCD in FLINT



- over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{F}_p$, $\mathbb{F}_{p^n}$ for $p < 2^{64}$.
- Lex, degLex and degGrevLex ordering supported
- Sequential alg: improves `Singular`'s
- Close to linear scaling
- `Singular` now relies on `FLINT`

New Perspectives and directions

Parallelizing memory intensive kernels

- Overhead of split and combine
- Only amortized for large instances
- Parallelization at a coarser grain

New Perspectives and directions

Parallelizing memory intensive kernels

- Overhead of split and combine
- Only amortized for large instances
- Parallelization at a coarser grain

Data placement

- Heavy use of dynamic allocation (`malloc`)
- Causes performances fickle
- To be used with parsimony
- Investigate dedicated allocators

New Perspectives and directions

Parallelizing memory intensive kernels

- Overhead of split and combine
- Only amortized for large instances
- Parallelization at a coarser grain

Data placement

- Heavy use of dynamic allocation (`malloc`)
- Causes performances fickle
- To be used with parsimony
- Investigate dedicated allocators

New directions

Application driven:

- More orderings: block, weighted
- Factorization: (harnessing most T5.4 contributions)

Conclusion

Outcome of WorkPackage 5

	Review 1	Review 2	Final review
T5.1 Pari/GP			D5.16
T5.2 GAP			D5.15
T5.3 LinBox		D5.12	D5.14
T5.4 Singular	D5.6, D5.7		D5.13
T5.5 MPIR	D5.5, D5.7		
T5.6 Combinatorics	D5.1	D5.11	
T5.7 Pythran	D5.2	D5.11	
T5.8 SunGrid Engine	D5.3		

Overall

- 31 software relases
- 16 research papers in journals or conference proceedings

Lessons learnt

From dedicated to general purpose HPC components:

- Early instances of HPC computer algebra: dedicated to some target application (breaking RSA, etc)
- Building a general purpose HPC component:
  - challenging
  - longer term sustainability
  - integration/composition of parallel components

Identifying the right place to focus efforts on

- Premature focus on embarassingly parallel codes may be an error

Risk of technology dependency

- Cilk: from success to shut-down
- Interchangeability and modularity (PARI, LinBox)

Perspectives

Interractive control over the architecture at the VRE level

- threads per component, GPUs, distributed nodes

➔ user decision based on algorithms, not systems

➔ slick interface of the VRE vs. compilation hurdle

Parallelism friendly portable containers

- supporting SIMD, multicores, accelerators

Exploiting emerging technologies:

- Non-Volatile RAM:
  - ➔ cheaper fat nodes,
  - ➔ but deeper cache hierarchy