Workpackage 5:
High Performance Mathematical Computing

Vincent Delecroix,
Alexander Konovalov,
Clément Pernet,

Second OpenDreamKit Project review

Luxembourg, October 30, 2018

High performance mathematical computing

Mathematical computing

Computing with a large variety of objects

- $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q,$ 17541718814389012164632

for applications where all digits matter.

High performance mathematical computing

Mathematical computing

Computing with a large variety of objects

- $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q$, $\quad 1754171881438901216463 2$
- Polynomials over $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q$, $\quad \frac{2}{5}x^3 + x^2 - \frac{1}{19}x + 2$

for applications where all digits matter.

High performance mathematical computing

Mathematical computing

Computing with a large variety of objects

- $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q,$ $\quad 1754171881438901216463\,2$
- 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 $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q,$ $\quad \begin{bmatrix} 27 & 3 & -1 \\ 9 & 0 & 2 \end{bmatrix}$

for applications where all digits matter.

High performance mathematical computing

Mathematical computing

Computing with a large variety of objects

- $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q$, $\quad 1754171881438901216463 2$
- 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 $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q$, $\quad \begin{bmatrix} 3x^2+3 & 2x^2+3 \\ 4x^2+1 & x^2+4x+4 \end{bmatrix}$

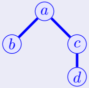
for applications where all digits matter.

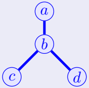
High performance mathematical computing

Mathematical computing

Computing with a large variety of objects

- $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q$, $\quad 175417188143890121646 32$
- 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 $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{F}_q$, $\quad \begin{bmatrix} 3x^2+3 & 2x^2+3 \\ 4x^2+1 & x^2+4x+4 \end{bmatrix}$

$\frac{3q^2-q^5}{q^5+2q^4+3q^3+3q^2+2q+1}$ 

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

- Tree algebras

for applications where all digits matter.

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 $\rightsquigarrow$ 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

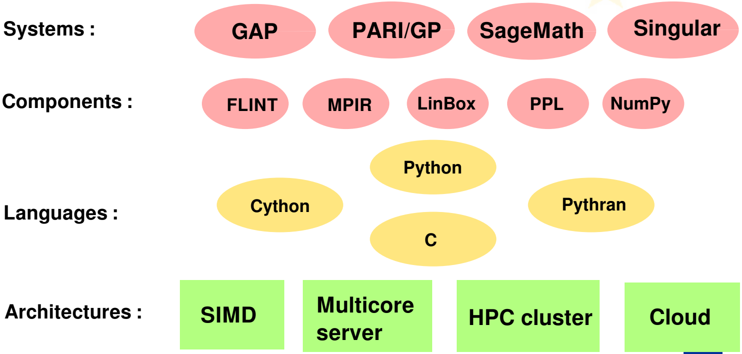
Architectures : SIMD, Multicore server, HPC cluster, Cloud

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)

$\rightsquigarrow$ 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

Milestone M8: Seamless use of parallel computing architecture in the VRE (proof of concept)

Astrid wants to run compute intensive routines involving both dense linear algebra and combinatorics. She has access through a JupyterHub-based VRE to a high end multi-core machine which includes a vanilla SAGE installation. She automatically benefits from the HPC features of the underlying specialized libraries (LinBox, ...). This is a proof of concept of the overall framework to integrate the HPC advances of specialized libraries into a general purpose VRE. It will prepare the final integration of a broader set of such parallel features for the end of the project

Outline

Workpackage management

Exact linear algebra

- Exact linear algebra algorithms and implementations (D5.12).
- Distributed computing

Combinatorics

- Refactor and optimize Sage's Combinatorics (D5.11)

Progress report on other tasks

- T5.1: Pari
- T5.2: GAP
- T5.4: Singular

Outline

Workpackage management

Exact linear algebra

- Exact linear algebra algorithms and implementations (D5.12).
- Distributed computing

Combinatorics

- Refactor and optimize Sage's Combinatorics (D5.11)

Progress report on other tasks

- T5.1: Pari
- T5.2: GAP
- T5.4: 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

- 20+ software releases
- 7 research papers

Addressing recommendations of review 1

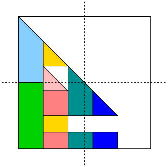
Recommendation 10: *Regarding WP5,* **make contacts** *with HPC community in order to ascertain current state-of-the-art. The work in this WP needs to be* **nearer the leading edge***.*

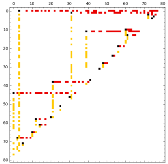
Addressing recommendations of review 1

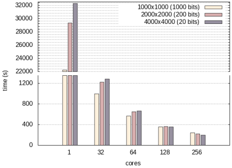
Recommendation 10: *Regarding WP5,* **make contacts** *with HPC community in order to ascertain current state-of-the-art. The work in this WP needs to be* **nearer the leading edge***.*

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







Strengthening interactions with numerical HPC community

Existing connection

Dense linear algebra numerical BLAS used for exact FFLAS for 17 years

Pointwise interactions: J. Dongarra, L. Grigori, J-Y. L'Excellent, etc

Publications to main HPC venues: SIAM-PPSC, EuroPar, PMAA, ParCo

Animation: involved in the French CNRS *Calcul* working group (Sci. Comp.)

Strengthening interactions with numerical HPC community

Existing connection

Dense linear algebra numerical BLAS used for exact FFLAS for 17 years

Pointwise interactions: J. Dongarra, L. Grigori, J-Y. L'Excellent, etc

Publications to main HPC venues: SIAM-PPSC, EuroPar, PMAA, ParCo

Animation: involved in the French CNRS *Calcul* working group (Sci. Comp.)

Recently established

- With the `BLIS` group:
  - Reporting bugs (SIMD vectorization)
  - Sharing experience in implementing Strassen's algorithm
- On-going collaboration with T. Mary (`Mumps`) and S. Chandrasekaran (UCSB) on quasiseparable matrix algorithmic

Outline

Workpackage management

Exact linear algebra

- Exact linear algebra algorithms and implementations (D5.12).
- Distributed computing

Combinatorics

- Refactor and optimize Sage's Combinatorics (D5.11)

Progress report on other tasks

- T5.1: Pari
- T5.2: GAP
- T5.4: Singular

Task 5.3: LinBox, High performance exact linear algebra

Mathematics is the art of reducing any problem to linear algebra

– W. Stein

Task 5.3: LinBox, High performance exact linear algebra

Mathematics is the art of reducing any problem to linear algebra
– W. Stein

Linear algebra: a key building block for HPC

Similarities with numerical HPC

- central elementary problem to which others reduce to
- (rather) simple algorithmic
- high compute/memory intensity

Task 5.3: LinBox, High performance exact linear algebra

Mathematics is the art of reducing any problem to linear algebra
– W. Stein

Linear algebra: a key building block for HPC

Similarities with numerical HPC

- central elementary problem to which others reduce to
- (rather) simple algorithmic
- high compute/memory intensity

Specificities

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

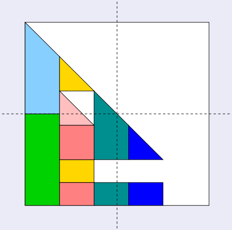
D5.12: Exact linear algebra algorithms and implementations. Library maintenance and close integration in mathematical software for LinBox library

1. Algorithmic innovations:
   1.1 Rank deficient dense Gaussian elimination
   1.2 Quasiseparable matrices
   1.3 Outsourced computing security
2. Software releases and integration:
   2.1 LinBox ecosystem: `LinBox`, `fflas-ffpack`, `givaro`
   2.2 `SageMath` integration

Rank deficient dense Gaussian elimination

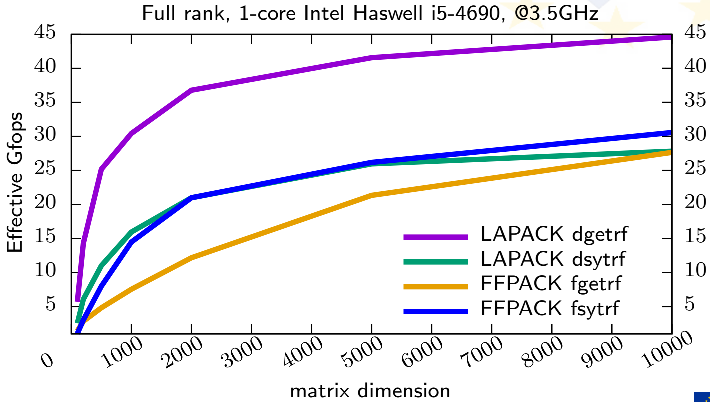
[ISSAC'18] Symmetric triangular factorization

- First unconditional recursive algorithm
- Pivoting revealing the Rank Profile Matrix
- $O(n^2 r^{\omega-2})$ ($= 1/3n^3$ with $\omega = 3, r = n$)
- Also hot topic in numerical linear algebra (LAPACK Working notes 294, Dec'17)



LAPACK vs FFPACK modulo 8 388 593

n	LAPACK (numerical) dgetrf (LU)	LAPACK (numerical) dsytrf (LDLT)	FFPACK (exact) fgetrf (LU)	FFPACK (exact) fsytrf (LDLT)
5000	2.01s	1.60s	3.90s	1.59s
10000	14.95s	11.98s	24.12s	10.90s

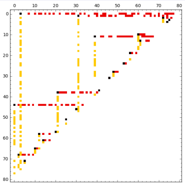


Quasiseparable matrices

Matrices with low off-diagonal rank

[ISSAC'16, JSC'18] New compact representation and algorithms

- Matches the best space complexities
- Reduction to matrix multiplication
- **Breakthrough:** flat representation (non hierarchical)

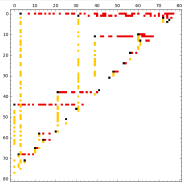


Quasiseparable matrices

Matrices with low off-diagonal rank

[ISSAC'16, JSC'18] New compact representation and algorithms

- Matches the best space complexities
- Reduction to matrix multiplication
- **Breakthrough:** flat representation (non hierarchical)



Follow-up: on-going collaboration with numerical HPC experts:

- S. Chandrasekaran (UCSB)
- T. Mary (U. Manchester, `Mumps`)

Software releases and integration

LinBox ecosystem

`givaro`: field/ring arithmetic

`fflas-ffpack`: dense linear algebra over finite field

`LinBox`: exact linear algebra

Tightly integrated in `SageMath`

Software releases and integration

LinBox ecosystem

`givaro`: field/ring arithmetic	4 releases
`fflas-ffpack`: dense linear algebra over finite field	6 releases
`LinBox`: exact linear algebra	6 releases
Tightly integrated in `SageMath`	13 tickets

Software releases and integration

LinBox ecosystem

`givaro`: field/ring arithmetic	4 releases
`fflas-ffpack`: dense linear algebra over finite field	6 releases
`LinBox`: exact linear algebra	6 releases
Tightly integrated in `SageMath`	13 tickets

Featuring

- Full functional implementations of new algorithmic contributions
- Improved vectorization and parallel routines
- Drastic improvement of reliability (continuous integration, test-suite coverage, randomized certificates, etc)

Distributed computing (on-going work)

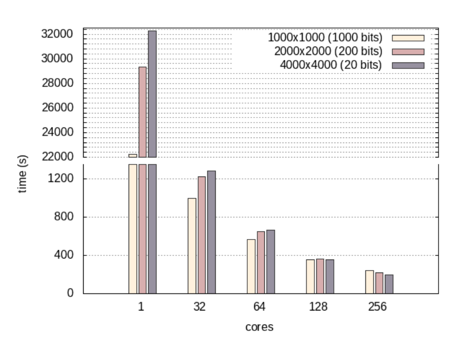
D5.14: Distributed exact linear system solving

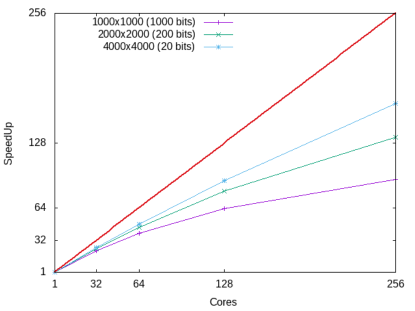
- 2 full time engineers
- Communication and serialization layer done
- Prototype MPI parallelization of Chinese remainder based solver.

Distributed computing (on-going work)

D5.14: Distributed exact linear system solving

- 2 full time engineers
- Communication and serialization layer done
- Prototype MPI parallelization of Chinese remainder based solver.



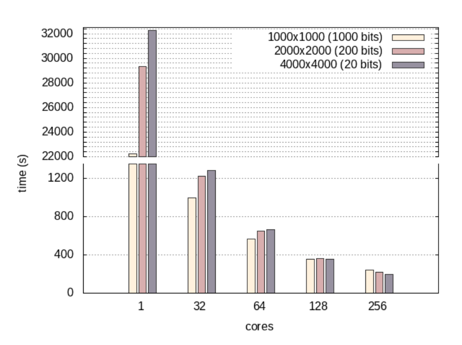


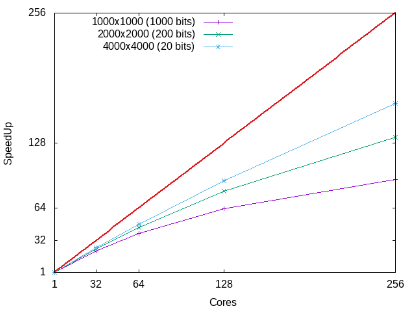
Distributed computing (on-going work)

D5.14: Distributed exact linear system solving

Roadmap:

- Major refactorization of LinBox solver code under way
- Hybrid combination of CRT+Dixon and parallelization.
- Hyrbid OpenMP-MPI implementation





Outline

Workpackage management

Exact linear algebra

- Exact linear algebra algorithms and implementations (D5.12).
- Distributed computing

Combinatorics

- Refactor and optimize Sage's Combinatorics (D5.11)

Progress report on other tasks

- T5.1: Pari
- T5.2: GAP
- T5.4: Singular

Task 5.6: HPC infrastructure for Combinatorics

Parallel algebraic computations exhibit high degrees of irregularity, at multiple levels and make no use of floating-point operations. This combination means that symbolic computations are not suited to conventional HPC paradigms.

– P. Maier, D. Livesey, H.-W. Loidl, P. Trinder

Task 5.6: HPC infrastructure for Combinatorics

Parallel algebraic computations exhibit high degrees of irregularity, at multiple levels and make no use of floating-point operations. This combination means that symbolic computations are not suited to conventional HPC paradigms.

– P. Maier, D. Livesey, H.-W. Loidl, P. Trinder

General situation

Given a set S of combinatorial objects we want to

- find (or pick at random) an element of S
- list (or count) the elements of S

Task 5.6: HPC infrastructure for Combinatorics

Parallel algebraic computations exhibit high degrees of irregularity, at multiple levels and make no use of floating-point operations. This combination means that symbolic computations are not suited to conventional HPC paradigms.

– P. Maier, D. Livesey, H.-W. Loidl, P. Trinder

General situation

Given a set S of combinatorial objects we want to

- find (or pick at random) an element of S
- list (or count) the elements of S

Specificities of combinatorics

- **huge**: S does not fit in memory
- **embarassingly parallel**: $S = S_1 \cup \ldots \cup S_k$
- **unbalanced**: S_i sizes are highly unbalanced

D5.11: Refactor and optimise the existing combinatorics Sage code using the new developed Pythran and Cython features.

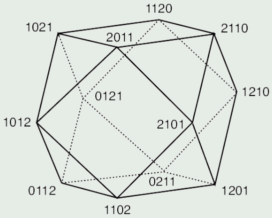
Algorithmic innovations and software integration through examples:

1. Example 1: polytopes
2. Example 2: numerical semigroups

Example 1: polytopes (description)

Example

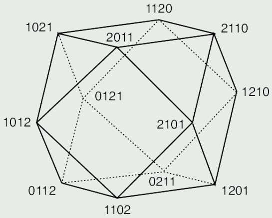
Some combinatorial objects can be encoded as integer vectors in polytopes.



Example 1: polytopes (description)

Example

Some combinatorial objects can be encoded as integer vectors in polytopes.



- Efficient algorithms available (*e.g.* Double Description, Barvinok)
- High performance libraries (*e.g.* `PPL`, `Normaliz`, `Polymake`)
- Useful for many combinatorial problems

Example 1: polytopes (ODK work)

ODK outcomes

`pplpy` library: Python interface to the high performance `PPL` (rational polytopes)

`e-antic` library: C/C++ library for exact computation over number fields. Building block for `Normaliz`.

Workshop: Sage Days 84

`SageMath`: improvement of native code and libraries interfaces

Example 1: polytopes (ODK work)

ODK outcomes

`pplpy` library: Python interface to the high performance `PPL` (rational polytopes) 6 releases

`e-antic` library: C/C++ library for exact computation over number fields. Building block for `Normaliz`. release soon

Workshop: Sage Days 84

`SageMath`: improvement of native code and libraries interfaces 10 tickets

Example 2: semigroups (description)

Example

Numerical semigroup are certain subsets of non-negative integers.

- $\langle 1 \rangle$
  - $\langle 2, 3 \rangle$
    - $\langle 2, 5 \rangle$
      - $\langle 2, 7 \rangle$
        - $\vdots$
    - $\langle 3, 4, 5 \rangle$
      - $\langle 3, 4 \rangle$
        - $\vdots$
      - $\langle 3, 5, 7 \rangle$
        - $\vdots$
      - $\langle 4, 5, 6, 7 \rangle$
        - $\vdots$

Example 2: semigroups (description)

Example

Numerical semigroup are certain subsets of non-negative integers.

- $\langle 1 \rangle$
  - $\langle 2, 3 \rangle$
    - $\langle 2, 5 \rangle$
      - $\langle 2, 7 \rangle$
        - $\vdots$
    - $\langle 3, 4, 5 \rangle$
      - $\langle 3, 4 \rangle$
        - $\vdots$
      - $\langle 3, 5, 7 \rangle$
        - $\vdots$
      - $\langle 4, 5, 6, 7 \rangle$
        - $\vdots$

- Many mathematical questions still open
- Highly unbalanced tree

Example 2: semigroups (ODK work)

A SageMath implementation of work-stealing map-reduce

- Work stealing algorithm (Leiserson-Blumofe)
- Easy to use, easy to call from SageMath with many use cases
- Scale well with the number of cores and reasonably efficient

Example 2: semigroups (ODK work)

A SageMath implementation of work-stealing map-reduce

- Work stealing algorithm (Leiserson-Blumofe)
- Easy to use, easy to call from SageMath with many use cases
- Scale well with the number of cores and reasonably efficient

A typical speedup (binary sequences)

# processors	1	2	4	8
Speedup	$\times 1$	$\times 1.55$	$\times 2.43$	$\times 2.87$

Example 2: semigroups (ODK work)

A SageMath implementation of work-stealing map-reduce

- Work stealing algorithm (Leiserson-Blumofe)
- Easy to use, easy to call from SageMath with many use cases
- Scale well with the number of cores and reasonably efficient

A typical speedup (binary sequences)

# processors	1	2	4	8
Speedup	$\times 1$	$\times 1.55$	$\times 2.43$	$\times 2.87$

integrated in SageMath

Example 2: semigroups (ODK work)

HPCombi: an experimental C++ library

- innovative SIMD code for combinatorics
- fine grained parallelization using Intel Cilk technology
- already in use for mathematical projects (numerical semigroups)

Example 2: semigroups (ODK work)

HPCombi: an experimental C++ library

- innovative SIMD code for combinatorics
- fine grained parallelization using Intel Cilk technology
- already in use for mathematical projects (numerical semigroups)

Operation	Speedup
Sum of a vector of bytes	$\times 3.81$
Sorting a vector of bytes	$\times 21.3$
Inverting a permutation	$\times 1.97$
Number of cycles of a permutation	$\times 41.5$
Number of inversions of a permutation	$\times 9.39$
Cycle type of a permutation	$\times 8.94$

Example 2: semigroups (ODK work)

HPCombi: an experimental C++ library

- innovative SIMD code for combinatorics
- fine grained parallelization using Intel Cilk technology
- already in use for mathematical projects (numerical semigroups)

Operation	Speedup
Sum of a vector of bytes	$\times 3.81$
Sorting a vector of bytes	$\times 21.3$
Inverting a permutation	$\times 1.97$
Number of cycles of a permutation	$\times 41.5$
Number of inversions of a permutation	$\times 9.39$
Cycle type of a permutation	$\times 8.94$

Open-source library

Outline

Workpackage management

Exact linear algebra

- Exact linear algebra algorithms and implementations (D5.12).
- Distributed computing

Combinatorics

- Refactor and optimize Sage's Combinatorics (D5.11)

Progress report on other tasks

- T5.1: Pari
- T5.2: GAP
- T5.4: Singular

T5.1: Pari

D5.16: Pari Suite release, fully supporting parallelization

- D5.10 (merged in D5.16):
  Generic parallelization engine is now mature (released since Nov 2016).
  Support POSIX-threads and MPI.
- Current work: applying it throughout the library
  - Chinese remaindering
  - Rational linear algebra
  - Discrete logarithm
  - Resultants
  - APRCL primality testing

T5.2: GAP

D5.15: Final report of GAP development

- 8 releases were produced integrating contributions of D3.11 and D5.15
- Towards an integration of HPC-GAP: main release GAP-4.9
  - Build system refactoring
  - Ability to compile in HPC-GAP compatibility mode
- Work in progress:
  - Multithreaded linear algebra: at the level of the `Meataxe` library
  - Introspection functionalities: on-the-fly optimisation decision

T5.4 Singular

D5.13: Parallel sparse polynomial multiplication

FLINT now supports fast sparse multivariate polynomials:

- addition, subtraction, multiplication,
- division, division with remainder, GCD
- evaluation, partial evaluation, composition

T5.4 Singular

D5.13: Parallel sparse polynomial multiplication

`FLINT` now supports fast sparse multivariate polynomials:

- addition, subtraction, multiplication,
- division, division with remainder, GCD
- evaluation, partial evaluation, composition

Parallelization of the (sparse) multiplication

threads	time (ms)	speedup
1	148661	x1.0
2	76881	x1.9
3	54798	x2.7
4	42855	x3.4
5	37017	x4.0
6	30892	x4.8
7	28365	x5.2
8	28048	x5.3

T5.4 Singular

D5.13: Parallel sparse polynomial multiplication

FLINT now supports fast sparse multivariate polynomials:

- addition, subtraction, multiplication,
- division, division with remainder, GCD
- evaluation, partial evaluation, composition

Parallelization of the (sparse) multiplication

threads	time (ms)	speedup
1	148661	x1.0
2	76881	x1.9
3	54798	x2.7
4	42855	x3.4
5	37017	x4.0
6	30892	x4.8
7	28365	x5.2
8	28048	x5.3

- Planned improvements to the memory manager $\Rightarrow$ closer to linear scaling
- Parallel division and GCD implementations are in progress.
- Integration into Factory/Singular remains to be done

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

- 20+ software relases
- 7 research papers

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

Risk of technology dependency

- Cilk: from success to shut-down
- Interchangeability and modularity

Extra Slides

Outsourced computing security (D5.12)

Exploratory aspect: Computation over the Cloud

Outsourcing computation on the cloud:

- trusted lightweight client computer
- untrusted powerful cloud server

$\Rightarrow$ need for certification protocols

Multiparty computation:

- each player contribute with a share of the input
- shares must remain private

Outsourced computing security (D5.12)

Exploratory aspect: Computation over the Cloud

Outsourcing computation on the cloud:

- trusted lightweight client computer
- untrusted powerful cloud server

$\Rightarrow$ need for certification protocols

Multiparty computation:

- each player contribute with a share of the input
- shares must remain private

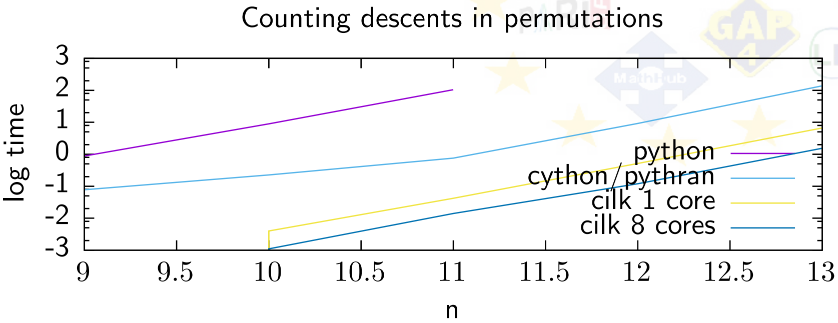
Contribution

ISSAC'17: Linear time certificates for LU, Det., Rank Profile matrix, etc

In submission: Secure multiparty Strassen's algorithm

Exploration: Python, Cython, Pythran and Cilk (D5.8)

workshop: Interfacing (math) software with low level libraries



- Python (interpreted language) slow.
- Cython/Pythran (Python to C compilers) efficient.
- Cilk (SIMD and multithread parallelism) very promising.