Hybrid Matrix Multiplication & Gaussian Elimination

Christian Eder, Jean-Charles Faugère, Fayssal Martani
and Bjarke Hammersholt Roune

HPAC Meeting – Lyon

December 10 – 11, 2013

- Fast linear algebra for computing Gröbner bases
- Hybrid Matrix Multiplication
- Next steps on matrix multiplication
- First steps on Gaussian Elimination
- Next steps

First attempts

In 2011 the original LELA project for fast linear algebra on special structured hybrid-sparse-dense matrices was started at the university of Kaiserslautern in the Singular team by Bradford Hovinen.

`https://github.com/Singular/LELA`

First attempts

In 2011 the original LELA project for fast linear algebra on special structured hybrid-sparse-dense matrices was started at the university of Kaiserslautern in the Singular team by Bradford Hovinen.

`https://github.com/Singular/LELA`

In 2012, doing a master thesis at the INRIA PolSys team Fayssal Martani optimized this code and added a completely new component with a specialized algorithm for computing the Gaussian Elimination.

`https://github.com/martani/LELA`

First attempts

In 2011 the original LELA project for fast linear algebra on special structured hybrid-sparse-dense matrices was started at the university of Kaiserslautern in the Singular team by Bradford Hovinen.

`https://github.com/Singular/LELA`

In 2012, doing a master thesis at the INRIA PolSys team Fayssal Martani optimized this code and added a completely new component with a specialized algorithm for computing the Gaussian Elimination.

`https://github.com/martani/LELA`

Also in 2012, Bjarke Hammersholt Roune started working on Linear Algebra for computing Gröbner bases in the MATHICGB package at the university of Kaiserslautern.

`https://github.com/broune/mathicgb`

Basic idea

The idea for a specialized Gaussian Elimination was initially presented by Faugère and Lachartre ([3]).

Basic idea

The idea for a specialized Gaussian Elimination was initially presented by Faugère and Lachartre ([3]).

- Special structure of the matrices due to Gröbner basis computations.

Basic idea

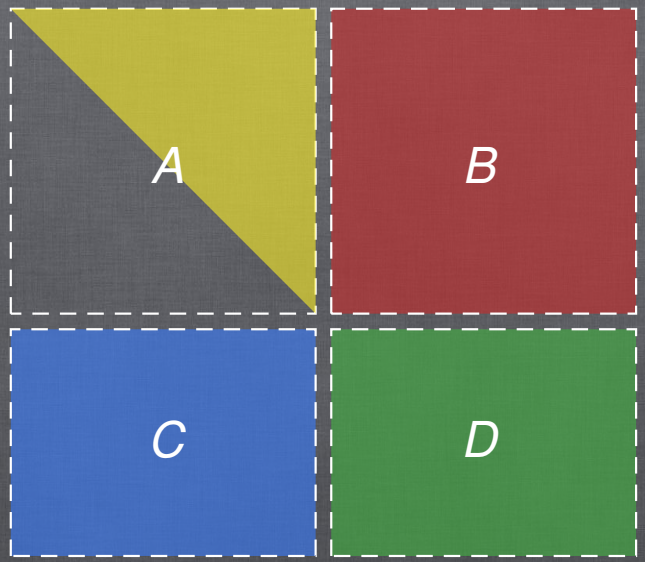
The idea for a specialized Gaussian Elimination was initially presented by Faugère and Lachartre ([3]).

- Special structure of the matrices due to Gröbner basis computations.
- Restructure matrix, swap rows and columns, take care of underlying monomial order

Basic idea

The idea for a specialized Gaussian Elimination was initially presented by Faugère and Lachartre ([3]).

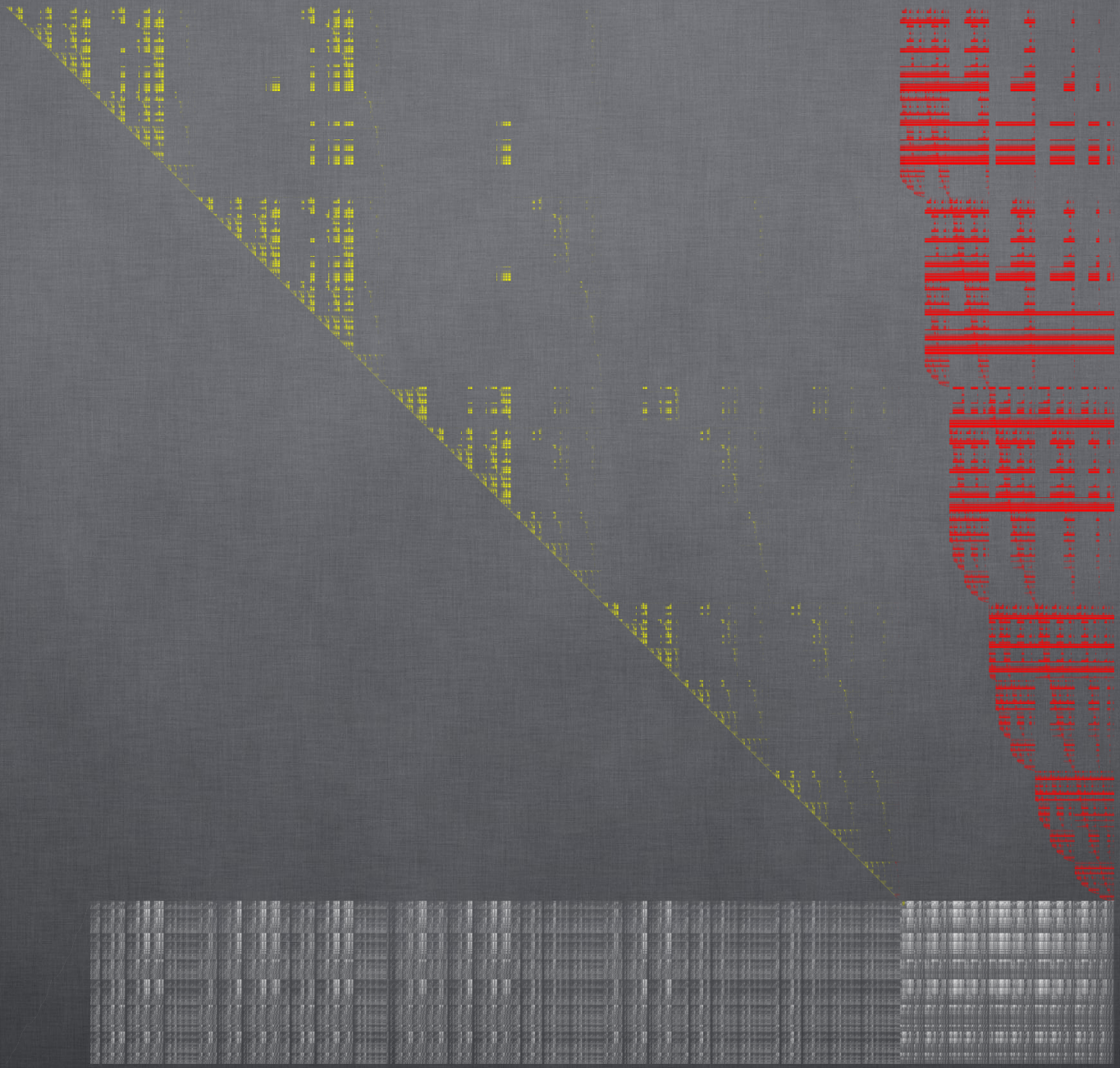
- Special structure of the matrices due to Gröbner basis computations.
- Restructure matrix, swap rows and columns, take care of underlying monomial order
- Split matrix in 4 parts:



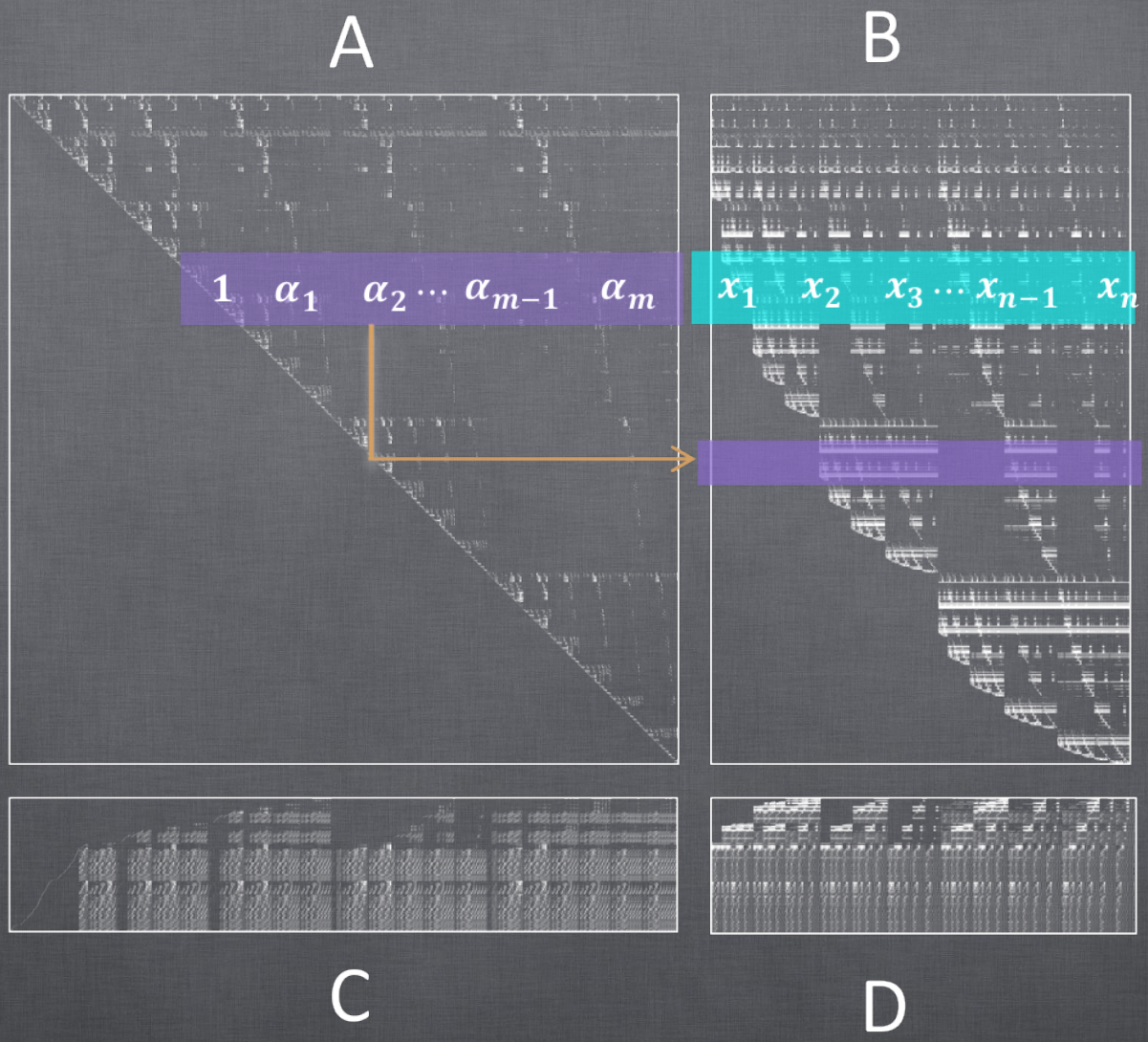
How our matrices look like

- Fast linear algebra for computing Gröbner bases
- Hybrid Matrix Multiplication
- Next steps on matrix multiplication
- First steps on Gaussian Elimination
- Next steps

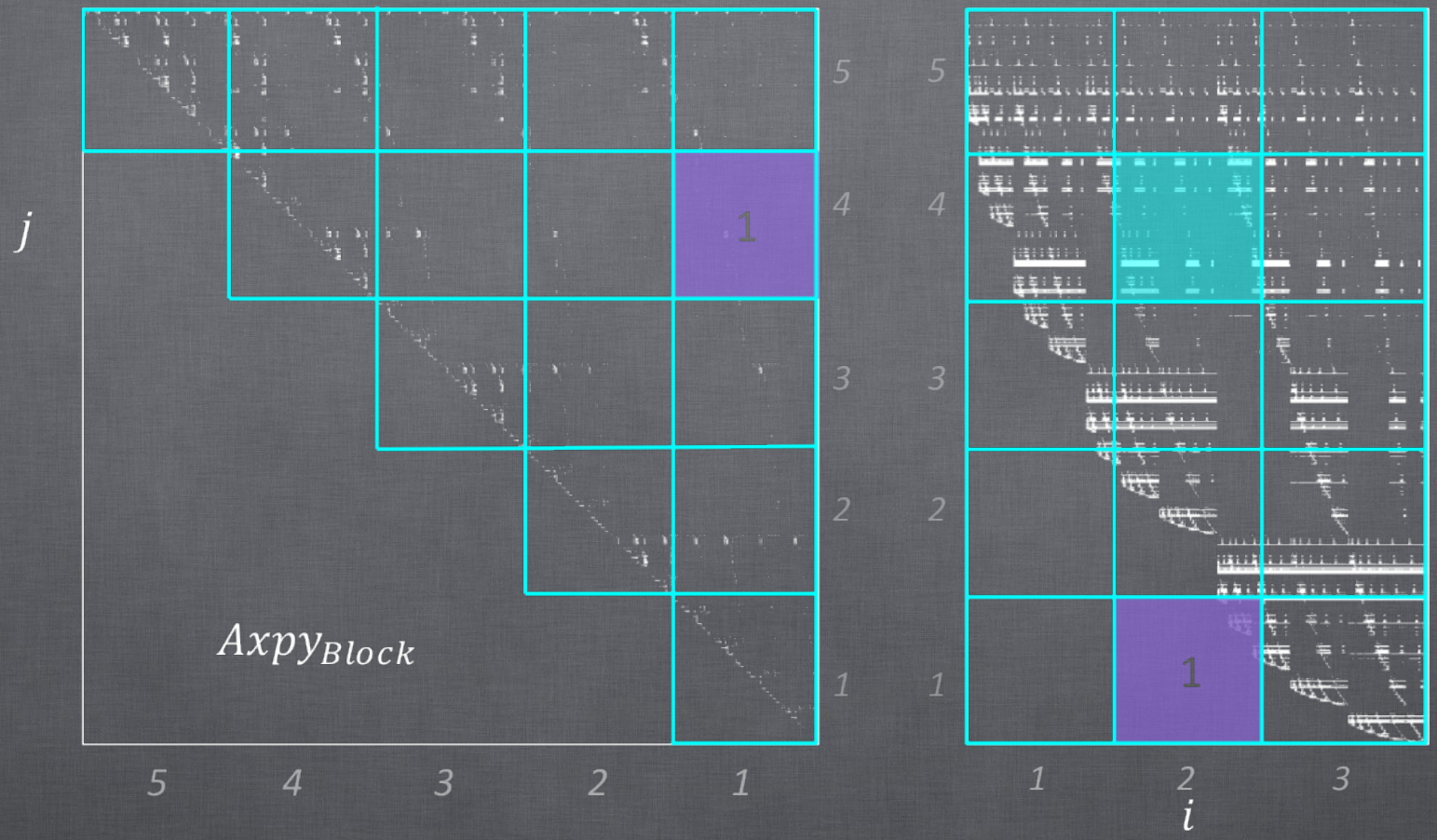
Hybrid Matrix Multiplication



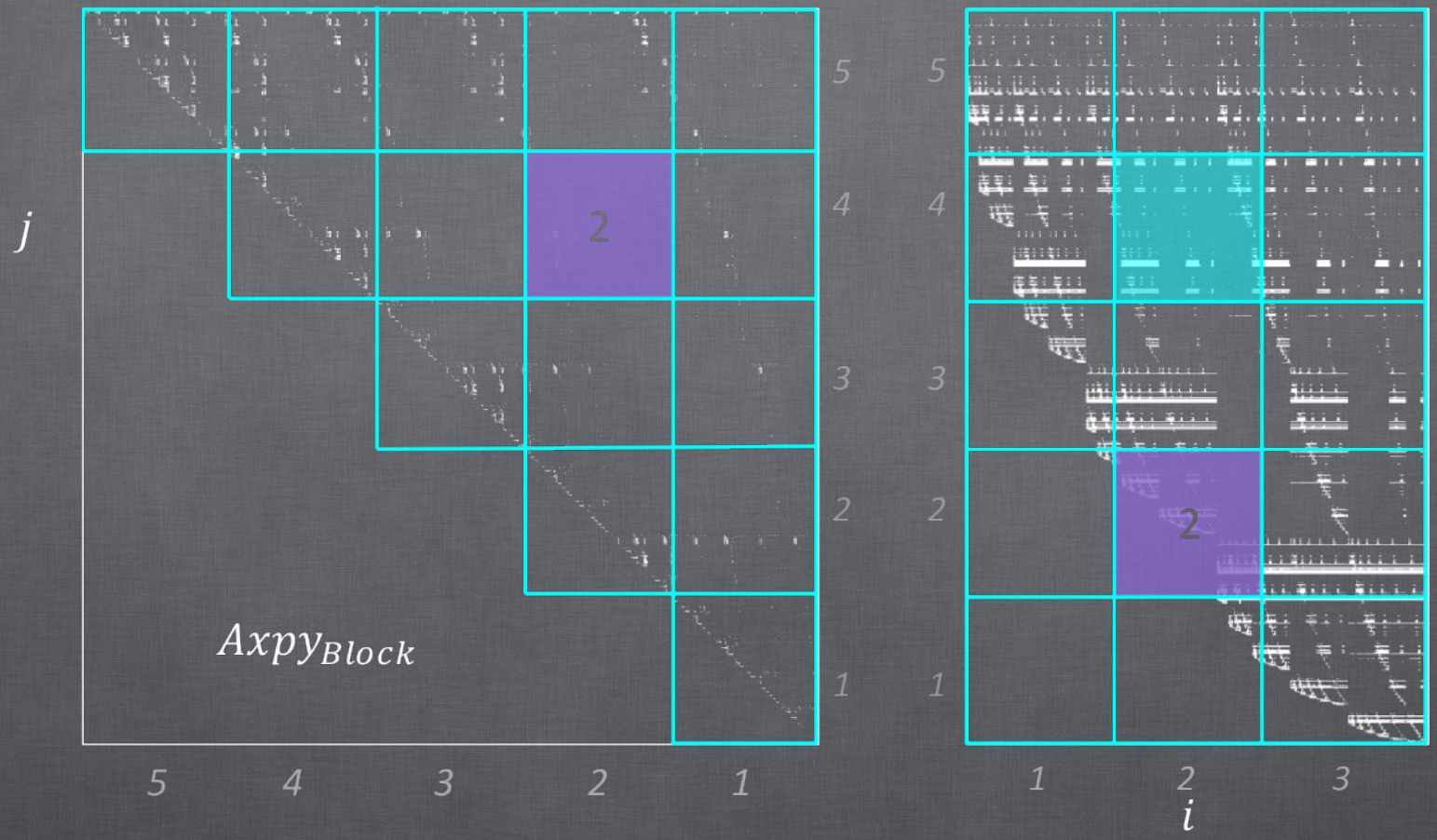
$$B \leftarrow A^{-1}B$$



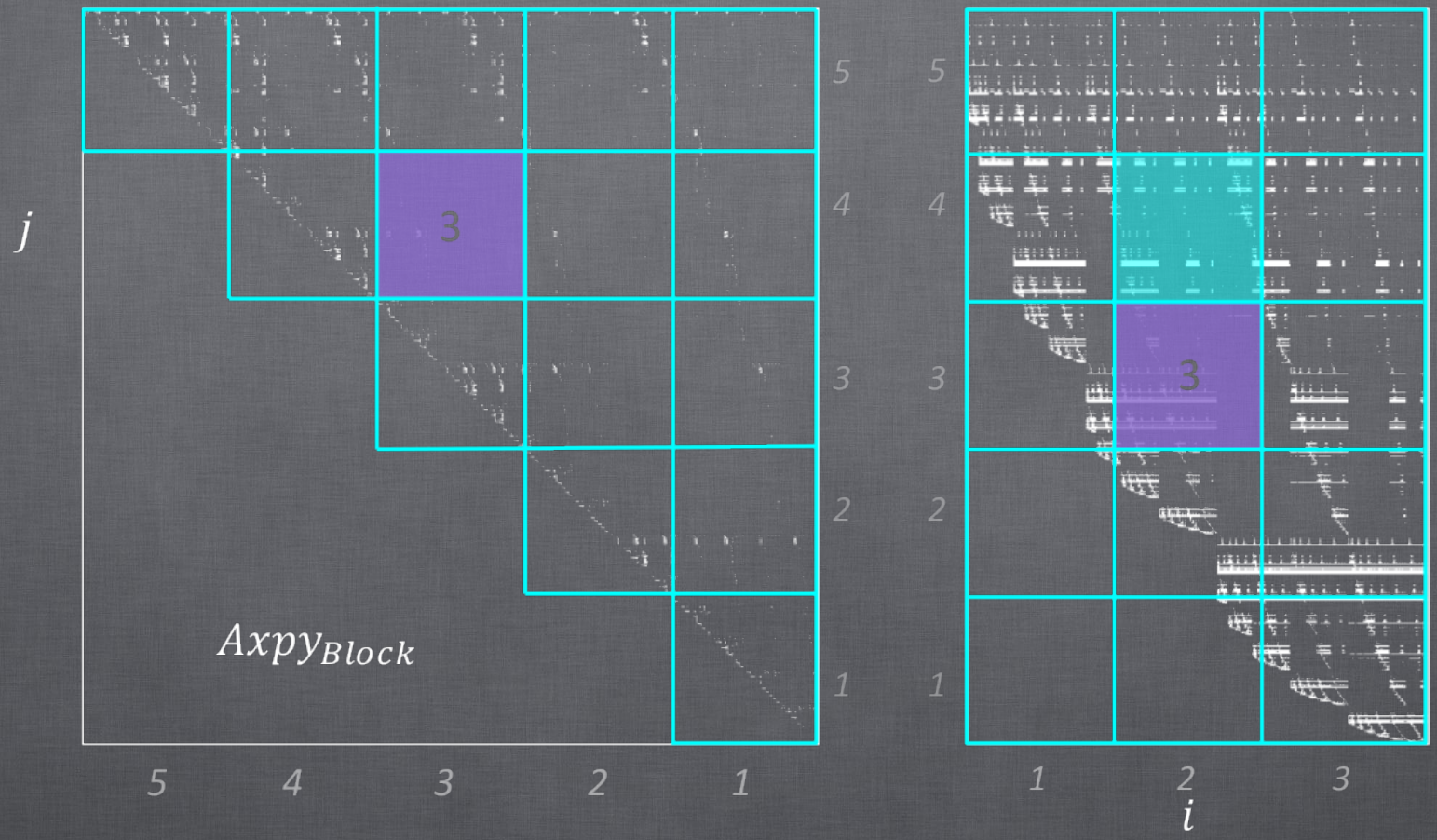
$B \leftarrow A^{-1}B$ – Block Version



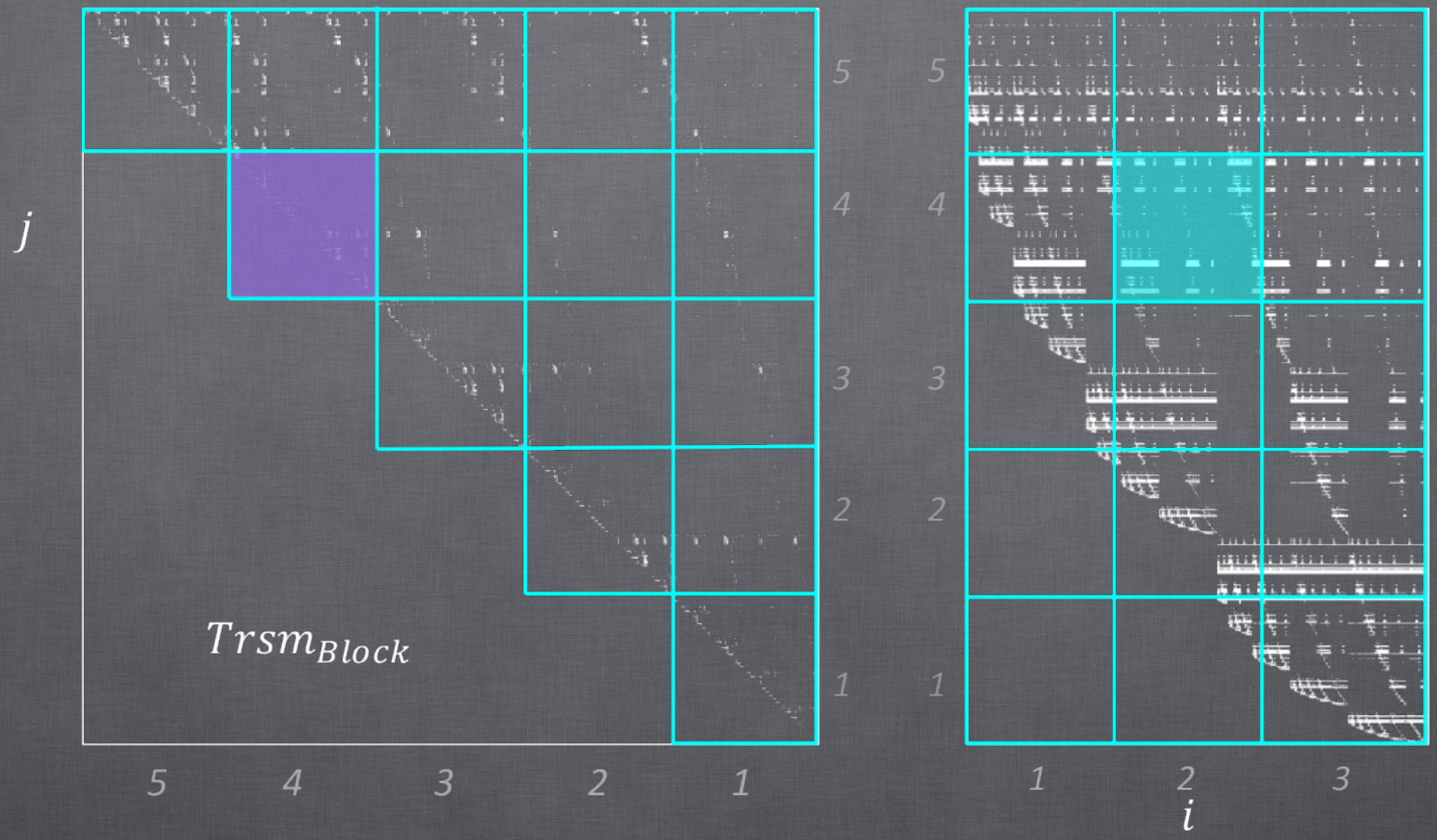
$B \leftarrow A^{-1}B$ – Block Version



$B \leftarrow A^{-1}B$ – Block Version



$B \leftarrow A^{-1}B$ – Block Version



LELA implementation

```
1  #ifdef L1 //LEVEL 1 PARALLELISM
2  omp_set_nested(1);
3  #pragma omp parallel num_threads(NUM_THREADS_OMP_MASTER) {
4  #endif
5  //for all columns of blocs of B
6  #ifdef L1 //LEVEL 1 PARALLELISM
7  #pragma omp for schedule(dynamic) nowait
8  #endif
9  for (uint32 ii = 0; ii < nb_column_blocs_B; ++ii) {
10 //for all rows of blocs in A (starting from the end)
11 for (uint32 jj = 0; jj < nb_row_blocs_A; ++jj) {
12   const uint32 first_bloc_idx =
13     A.FirstBlocsColumIndexes[jj] / A.bloc_width();
14   const uint32 last_bloc_idx = MIN(A[jj].size () - 1, jj);
15
16   Level1Ops::memsetToZero(dense_bloc);
17   Level1Ops::copySparseBlocToDenseBlocArray(R, B[jj][ii], dense_bloc);
18
19   //for all the blocs in the current row of A
20   for (uint32 k = 0; k < last_bloc_idx; ++k) {
21     if (A[jj][k].empty() || B[k + first_bloc_idx][ii].empty())
22       continue;
23
24 #ifdef L2 //LEVEL 2!
25 #pragma omp parallel num_threads(NUM_THREADS_OMP_SLAVES_PER_MASTER) {
26 #pragma omp for schedule(dynamic) nowait
27 #endif
28     for (int i = 0; i < DEFAULT_BLOC_HEIGHT / 2; ++i) {
29       uint8 is_sparse = 0;
```

Some examples – Structures

Benchmark	size A	density A	size B	density B
Kat12-Mat7	$18,890 \times 18,890$	0.76%	$18,890 \times 4,237$	14.02%
Kat13-Mat10	$40,023 \times 40,023$	0.50%	$40,023 \times 8,204$	13.53%
Kat14-Mat14	$88,941 \times 88,941$	0.37%	$88,941 \times 16,397$	12.03%
Kat16-Mat6	$82,086 \times 82,086$	0.22%	$82,086 \times 32,166$	3.54%
Minrank-9-10-7-Mat3	$18,460 \times 18,460$	4.01%	$18,460 \times 56,095$	14.01%
Minrank-9-10-7-Mat4	$34,053 \times 34,053$	3.52%	$34,053 \times 74,125$	38.93%

Some examples – Timings LELA & MATHICGB

Benchmark	Timings in seconds with given number of threads						
	1	2	4	8	16	32	64
K12-M7-L	31.09	16.80	8.99	4.86	3.15	2.81	3.00
K14-M7-M	32.11	17.30	8.76	4.55	3.01	2.41	2.13
K13-M10-L	175.41	87.65	45.13	24.33	13.34	8.33	8.59
K13-M10-M	176.52	86.98	44.35	24.02	12.78	7.96	6.89
K14-M14-L	1,285.10	635.83	324.21	170.62	92.85	52.51	43.71
K14-M14-M	1,2799.23	631.74	322.56	168.75	93.45	50.13	38.68
K16-M6-L	593.32	298.87	153.33	79.17	45.66	29.62	37.59
K16-M6-M	597.41	300.16	154.67	78.53	44.77	27.32	31.14
M-9-10-7-M3-L	1,734.00	862.80	432.40	218.10	116.30	67.04	51.62
M-9-10-7-M3-M	1,745.81	867.98	438.31	221.12	124.46	73.01	58.23
M-9-10-7-M4-L	6,410.00	3,174.22	1,583.04	801.63	428.50	243.62	183.55
M-9-10-7-M4-M	6,421.23	3,197.42	1,612.87	809.62	426.16	247.11	188.16

Some examples – Speedups LELA & MATHICGB

Benchmark	Speedup w.r.t. computation on 1 thread						
	1	2	4	8	16	32	64
K12-M7-L	1.00	1.85	3.45	6.40	9.86	11.06	10.36
K12-M7-M	0.97	1.79	3.54	6.83	10.32	12.90	14.60
K13-M10-L	1.00	2.00	3.89	7.21	13.15	21.07	20.44
K13-M10-M	0.99	2.01	3.95	7.30	13.73	22.04	25.45
K14-M14-L	0.99	2.01	3.95	7.50	13.78	24.37	29.28
K14-M14-M	1.00	2.03	3.97	7.58	13.70	25.53	33.09
K16-M6-L	1.00	1.99	3.87	7.49	13.00	20.03	15.78
K16-M6-M	0.99	1.97	3.83	7.55	13.25	21.71	19.05
M-9-10-7-M3-L	1.00	2.01	4.01	7.95	14.91	25.87	33.59
M-9-10-7-M3-M	0.99	1.99	3.95	7.84	13.93	23.75	29.77
M-9-10-7-M4-L	1.00	2.02	4.04	8.00	14.96	26.31	34.93
M-9-10-7-M4-M	0.99	2.00	3.97	7.91	15.04	25.93	34.06

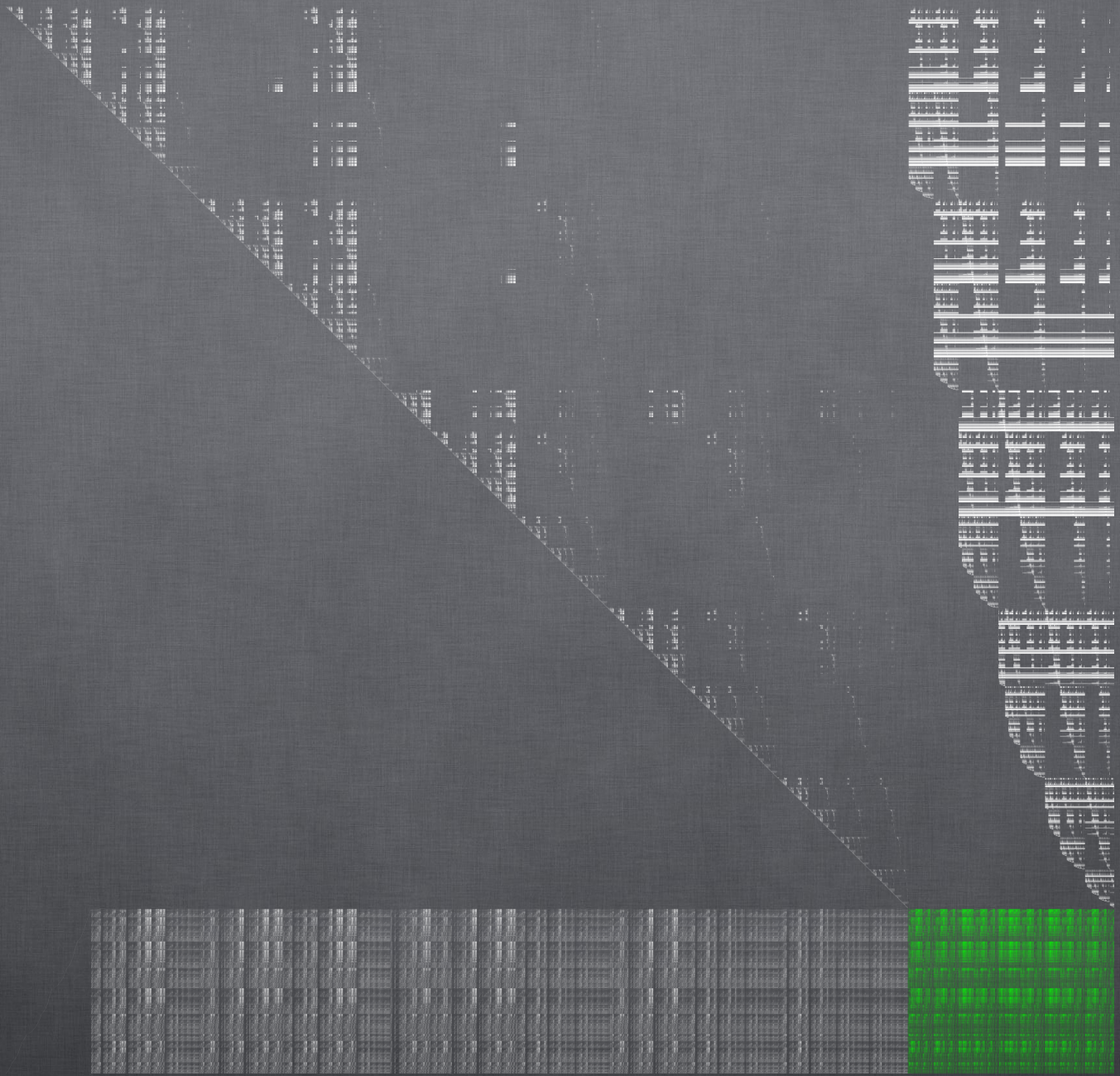
- Fast linear algebra for computing Gröbner bases
- Hybrid Matrix Multiplication
- **Next steps on matrix multiplication**
- First steps on Gaussian Elimination
- Next steps

Next steps

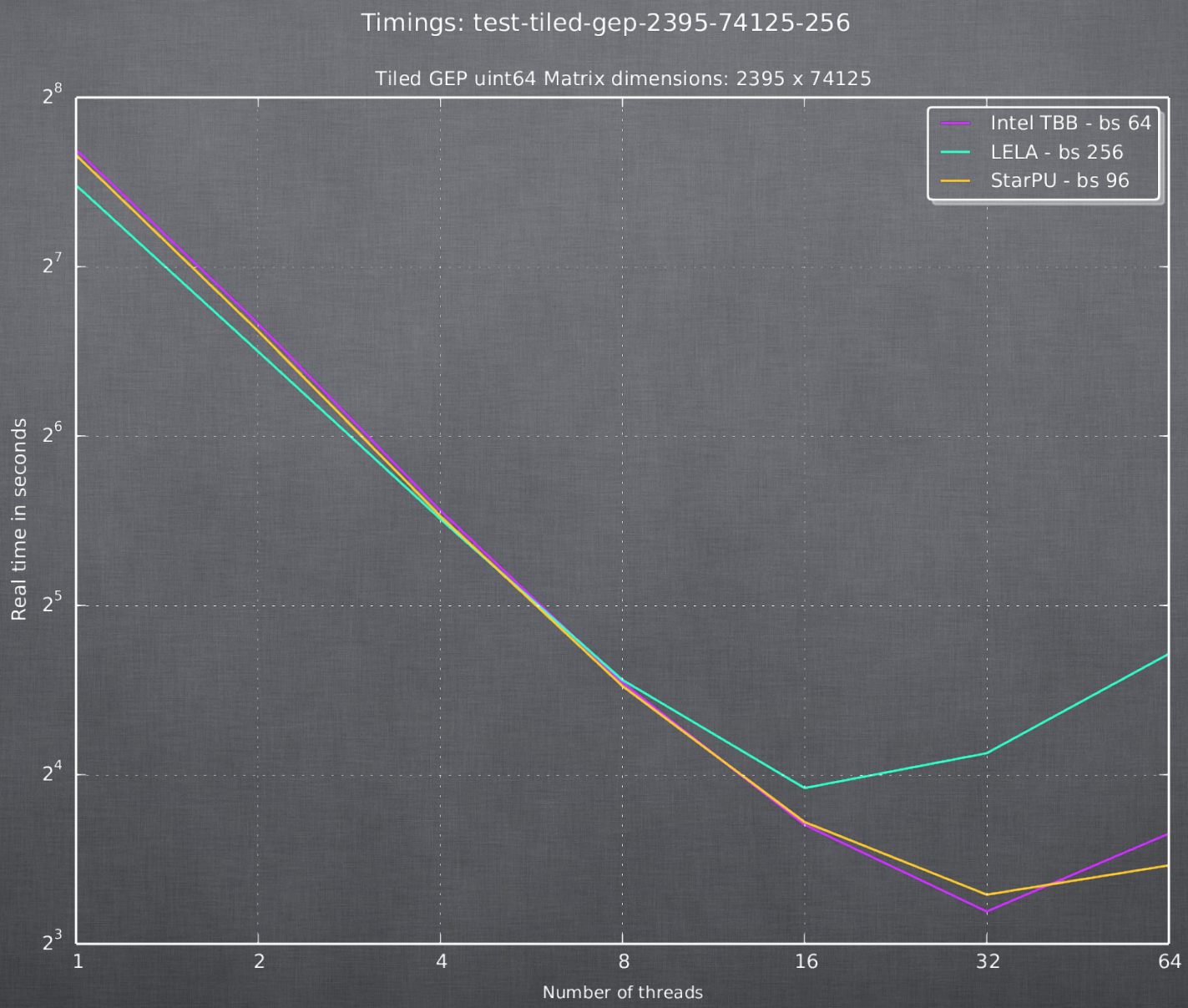
- Test and compare different sparse, hybrid, dense data structures in LELA and MATHICGB
- Implement and compare task-based hybrid tiled matrix multiplication with StarPU, OpenMP/KAAPI and Intel TBB

- Fast linear algebra for computing Gröbner bases
- Hybrid Matrix Multiplication
- Next steps on matrix multiplication
- First steps on Gaussian Elimination
- Next steps

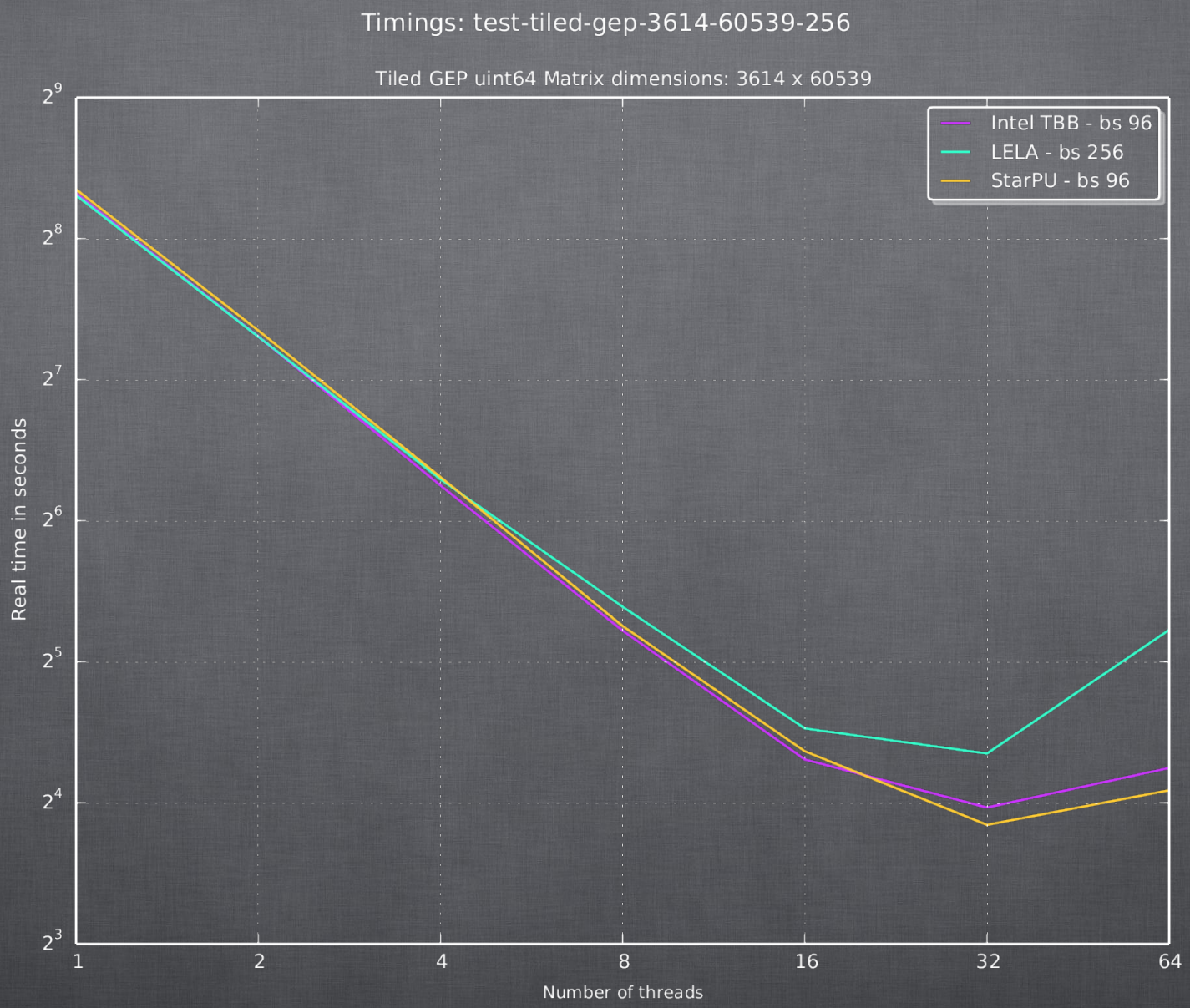
First step – Dense Gaussian Elimination



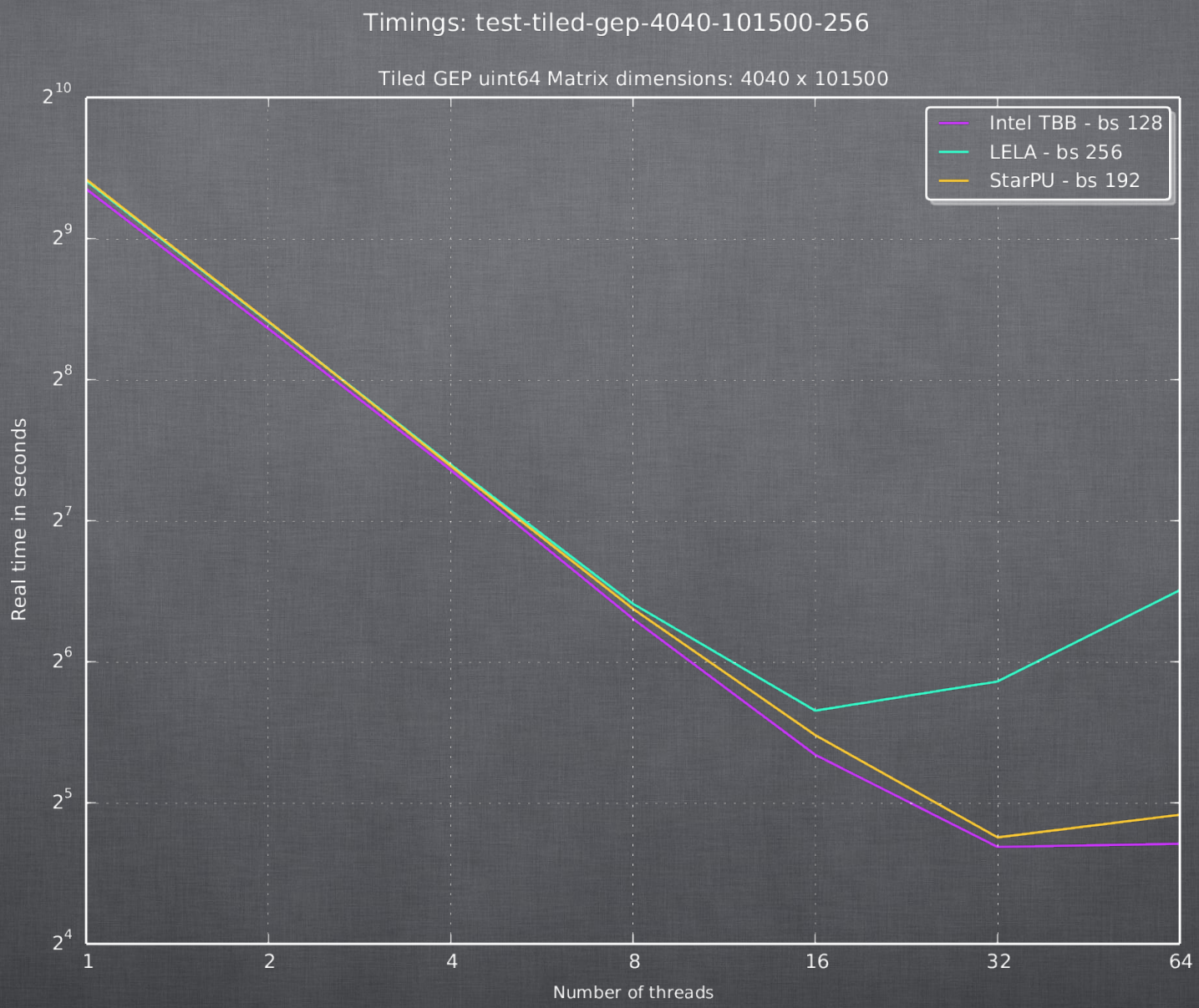
Some examples – Kat16-Mat9



Some examples – Minrank-10-9-7-Mat4



Some examples – Minrank-10-9-7-Mat6



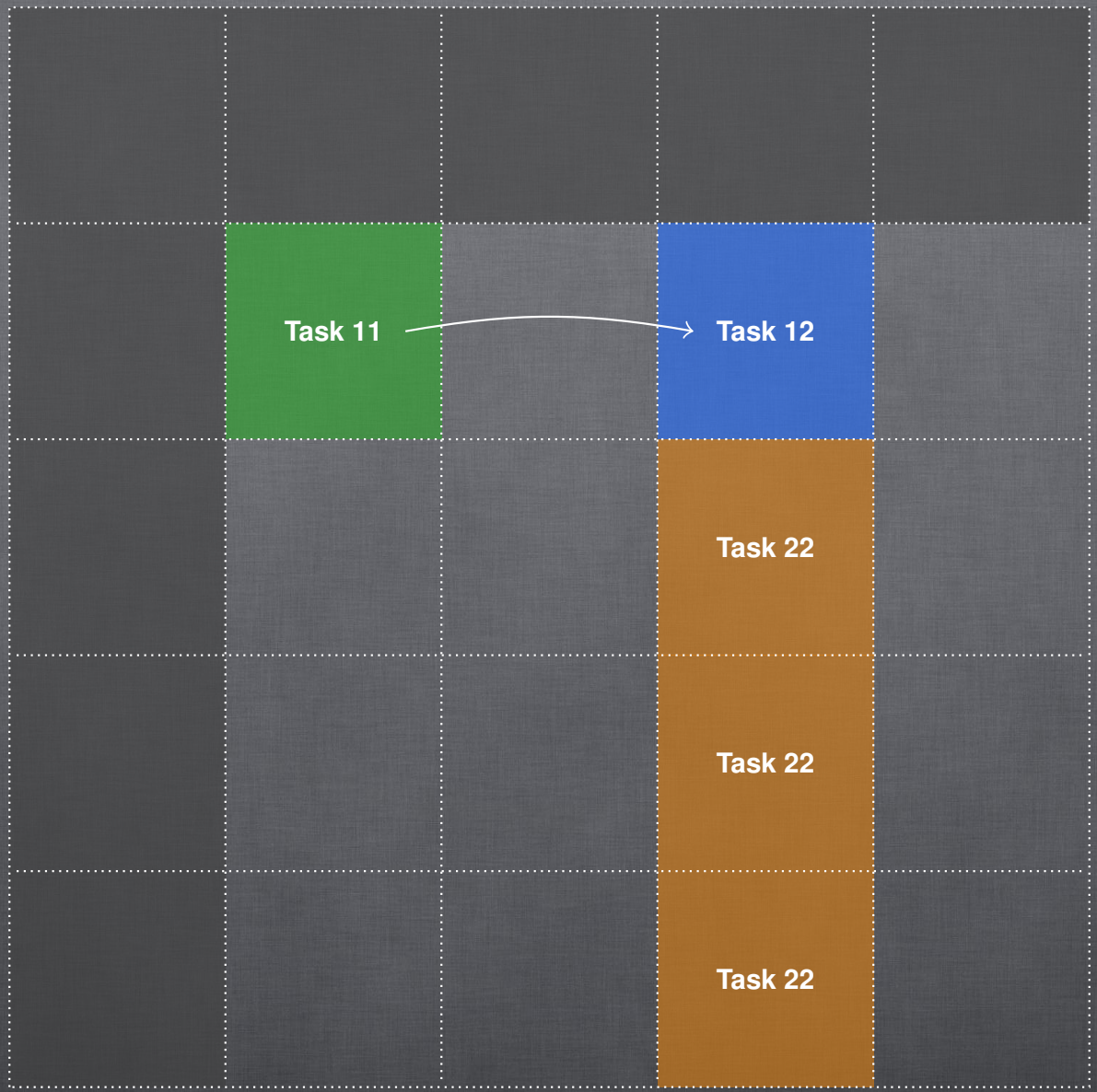
LELA implementation

- Using pthreads
- Idea of structured Gaussian Elimination:

$$\text{row}_i \leftarrow \text{row}_i + \sum_{j=1}^{i-1} a_j \text{row}_j$$

  At step i rows 1 to $i-1$ are reduced.
  - ▷ Reduce in parallel rows i to $i+p$ by pivots up to $i-1$
  - ▷ Row $i+\text{offset}$ waits until rows from i to $i+\text{offset}-1$ are done
    $\Rightarrow$ waiting queue
  - ▷ Threads share global variable `last_pivot`
- Blocksize 256 works out best for the shown examples

Tiled implementation



StarPU implementation

Tiled Gaussian Elimination

```
1  struct starpu_data_filter f = {
2    .filter_func = starpu_matrix_filter_vertical_block,
3    .nchildren = lblocks
4  };
5
6  struct starpu_data_filter f2 = {
7    .filter_func = starpu_matrix_filter_block,
8    .nchildren = mblocks
9  };
10
11 starpu_data_map_filters(dataA, 2, &f, &f2);
```

StarPU Implementation

Tasks & Dependencies

```
1  static int create_task_12(starpu_data_handle_t dataA, unsigned k,
       unsigned j)
2  {
3    int ret;
4
5    struct starpu_task *task = create_task(TAG12(k, j));
6
7    task->cl = &task_12_cl;
8
9    /* what sub-data is manipulated ? */
10   task->handles[0] = starpu_data_get_sub_data(dataA, 2, k, k);
11   task->handles[1] = starpu_data_get_sub_data(dataA, 2, k, j);
12
13   if (!no_prio && (j == k+1)) {
14     task->priority = STARPU_MAX_PRIO;
15   }
16
17   /* enforce dependencies ... */
18   if (k > 0) {
19     starpu_tag_declare_deps(TAG12(k, j), 2, TAG11(k), TAG22(k-1, k, j));
20   } else {
21     starpu_tag_declare_deps(TAG12(k, j), 1, TAG11(k));
22   }
23 ...
24 }
```

Intel TBB Implementation

```
class TASK_12 : public task {
public:
  TYPE *a;
  TYPE *b;
  TYPE offset;
  // successors, all TASK_22
  TASK_22 **task_22_succ;

  TASK_12(TYPE *a_, TYPE *b_, TYPE offset);

  task* execute();
};
```

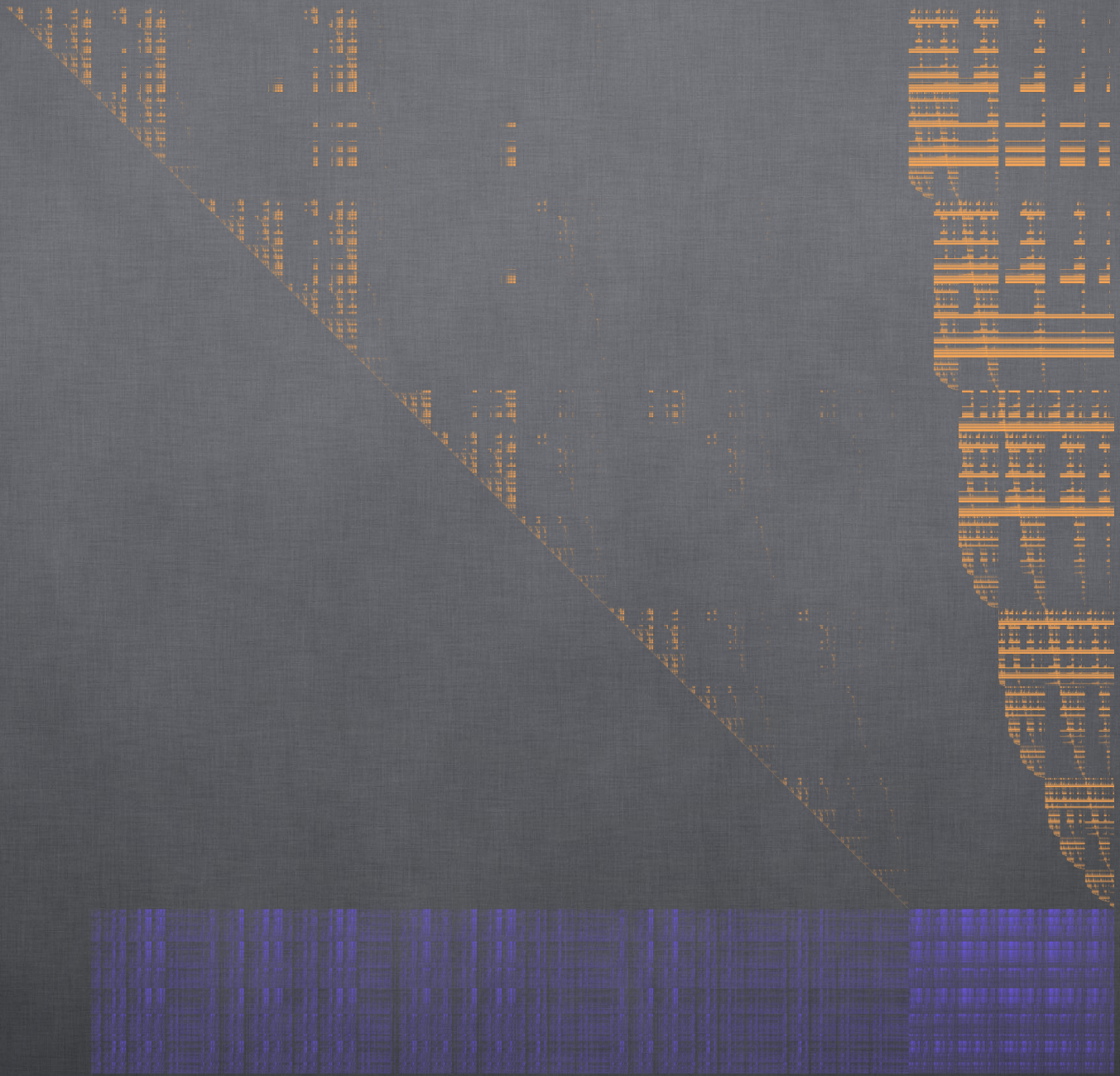
```
for (j=k+1; j<mblocks; ++j) {
  task_12_queue[task_12_ctr] = new(task::allocate_root())
    TASK_12(&mat[tile_size*(k+k*m)], &mat[tile_size*(j+k*m)],k);
  task_11_queue[k]->task_12_succ[j-k-1] = task_12_queue[task_12_ctr];
  if (k!=0) {
    task_22_queue[task_22_start_idx+j-k]->task_12_succ =
      task_12_queue[task_12_ctr];
    task_12_queue[task_12_ctr]->set_ref_count(2);
  } else {
    task_12_queue[task_12_ctr]->set_ref_count(1);
  }
  task_12_ctr++;
}
```

- Fast linear algebra for computing Gröbner bases
- Hybrid Matrix Multiplication
- Next steps on matrix multiplication
- First steps on Gaussian Elimination
- **Next steps**

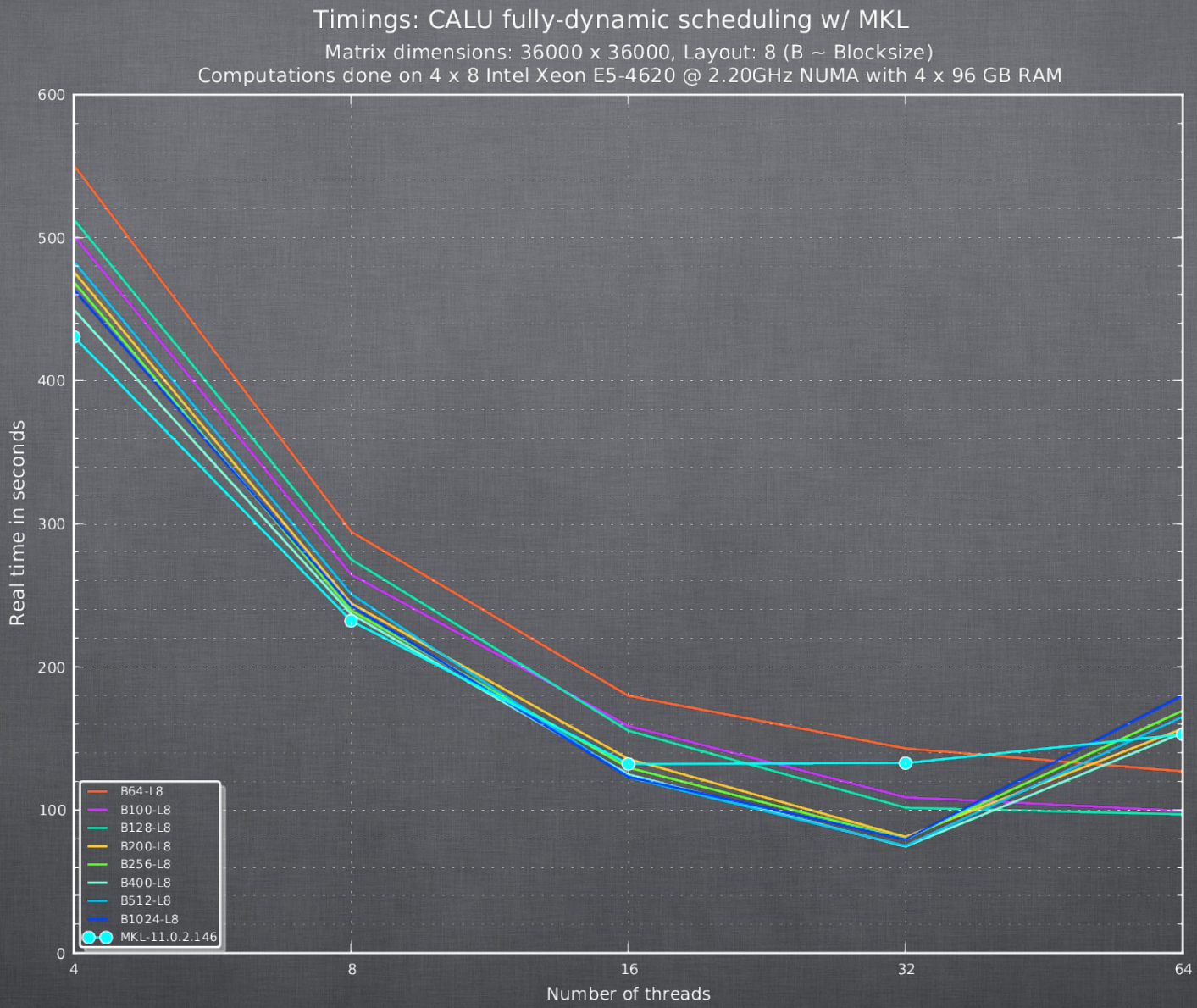
Next steps

- Finish Intel TBB implementation of dense Gaussian Elimination and compare to StarPU
- Improve scheduling for #columns $>>$ #rows
- Make these implementations available in LELA resp. MATHICGB

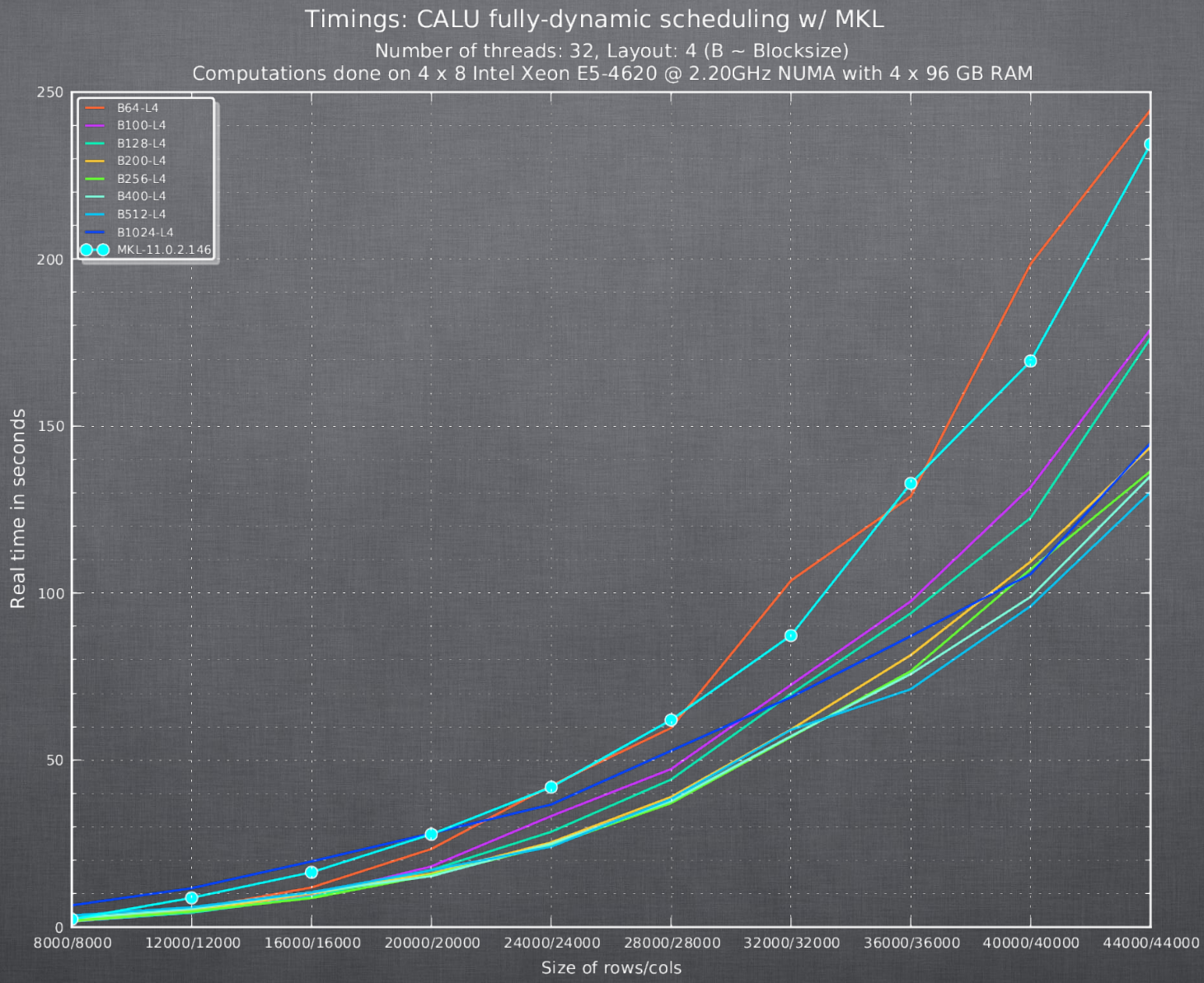
Directly reduce C?



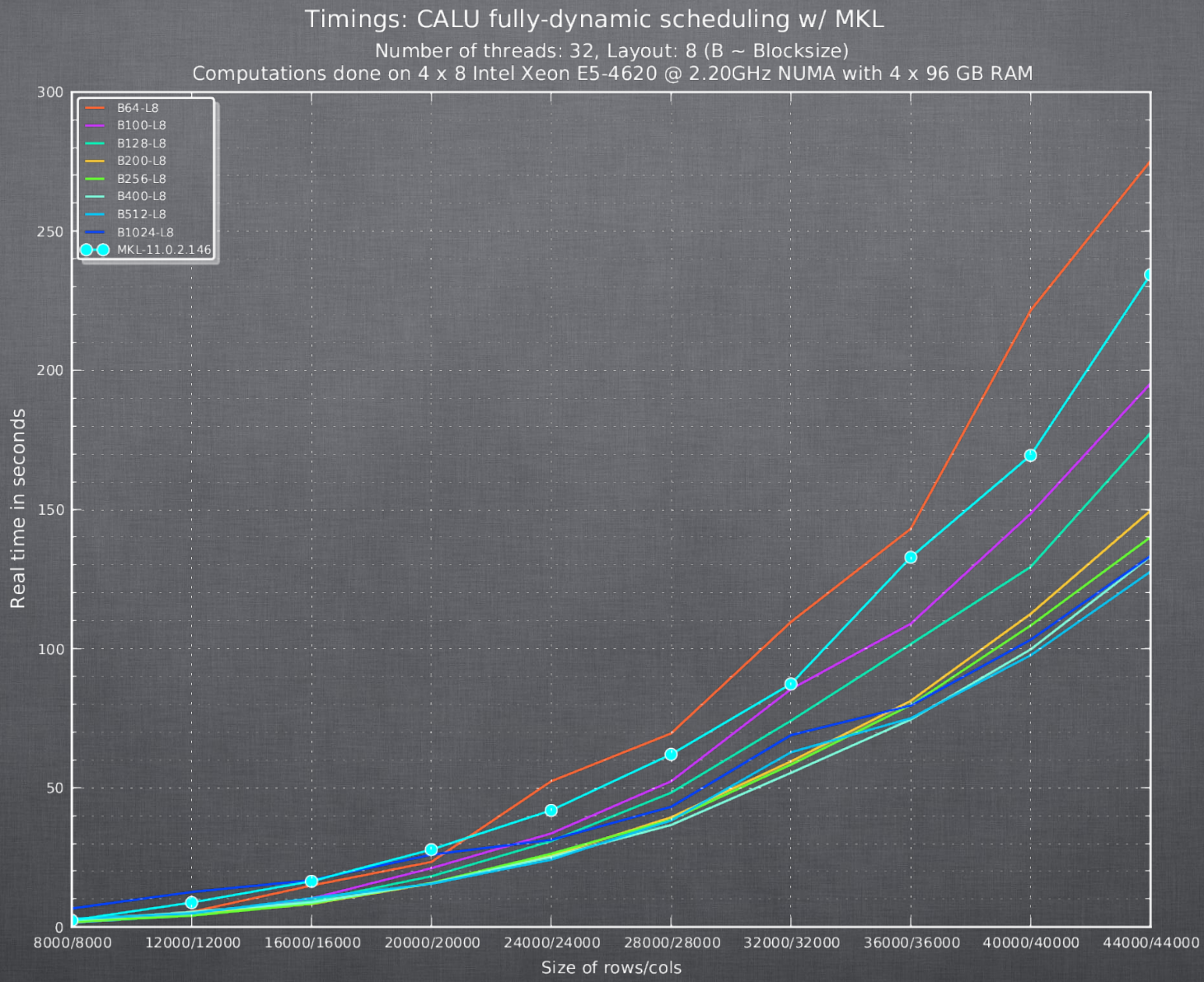
Other tests – CALU 1



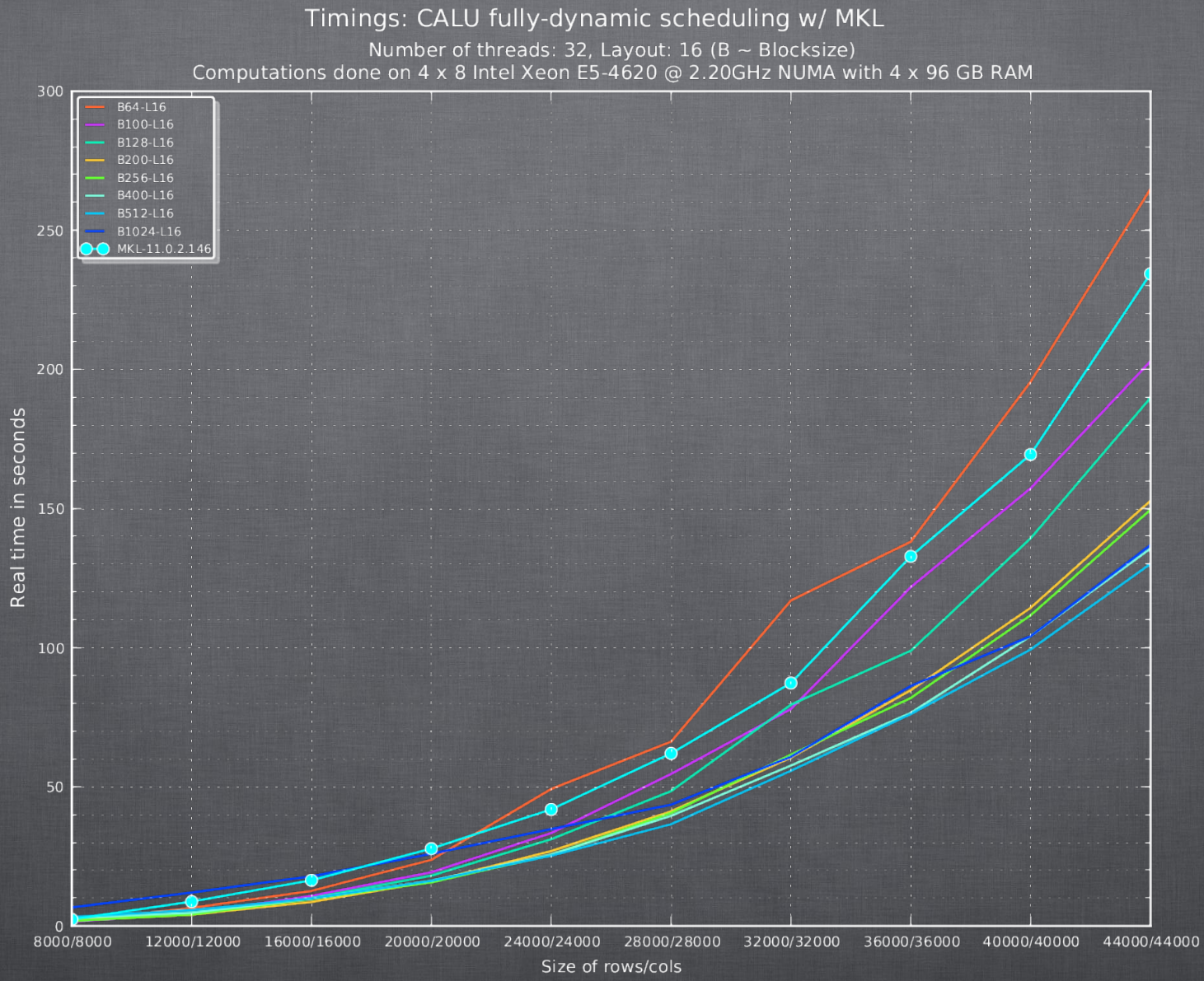
Other tests – CALU 2



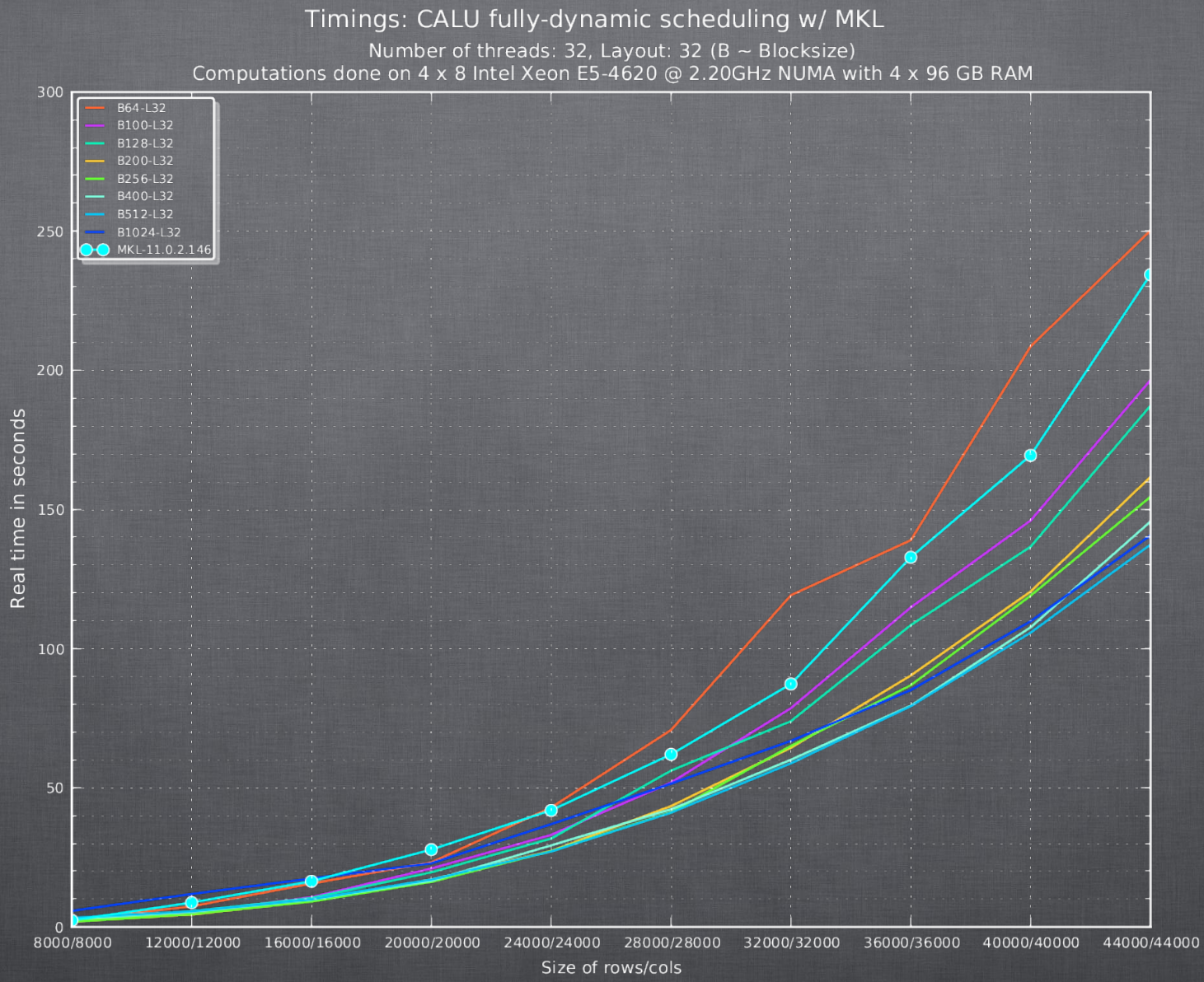
Other tests – CALU 2



Other tests – CALU 2



Other tests – CALU 2



References

[1] Demmel, J., Grigori, L., and Xiang, H. CALU: A Communication Optimal LU Factorization Algorithm. Technical Report UCB/EECS-2010-29, EECS Department, University of California, Berkeley, Mar 2010.

[2] Faugère, J.-C. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, June 1999. http://www-salsa.lip6.fr/~jcf/Papers/F99a.pdf.

[3] Faugère, J.-C. and Lachartre, S. Parallel Gaussian Elimination for Gröbner bases computations in finite fields. In M. Moreno-Maza and J.L. Roch, editor, *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, PASCO '10, pages 89–97, New York, NY, USA, July 2010. ACM.

[4] Lachartre, S. *Algèbre linéaire dans la résolution de systèmes polynomiaux, applications en cryptologie*. PhD thesis, L'université Pierre et Marie Curie – Paris VI, 2008.

[5] Martani, F. Faugère-Lachartre Parallel Gaussian Elimination for Gröbner Bases computations over finite fields – Implementations and new algorithms. Master's thesis, L'université Pierre et Marie Curie – Paris VI, 2012.