Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

LASs – Linear Algebra routines on OmpSs

Sandra Catalán (scatalan@bsc.es), Pedro Valero, Xavier Martorell and Jesús Labarta

Knoxville, April 15, 2019

9TH JLESC WORKSHOP

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

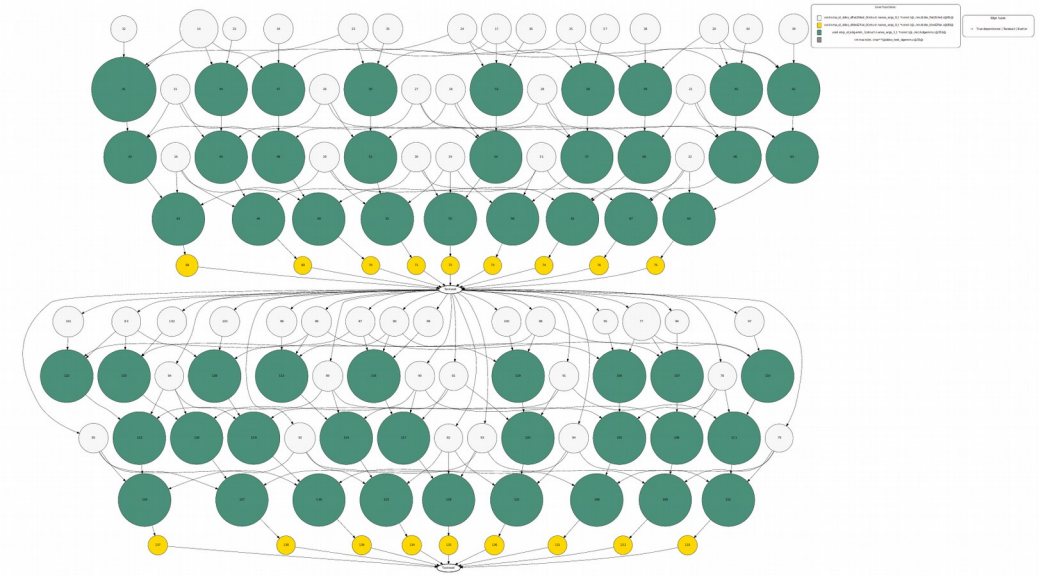
1) Introduction

2) TRSM in LASs
3) NPGETRF in LASs
4) sLASs on small mat.
5) SpMV
6) Ongoing work

Programing Model

OmpSs and OpenMP Tasks (Runtime):

- We divide the problem into tasks
  - Data dependences among task (DAG)
- Runtime manages and schedules the running of the tasks
  - Transparent
    - Runtime is in charge of the optimizations, not the programmer
  - Easy to implement
  - Easy to port
  - Easy to maintain

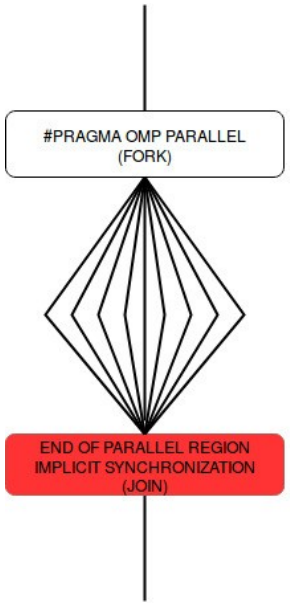


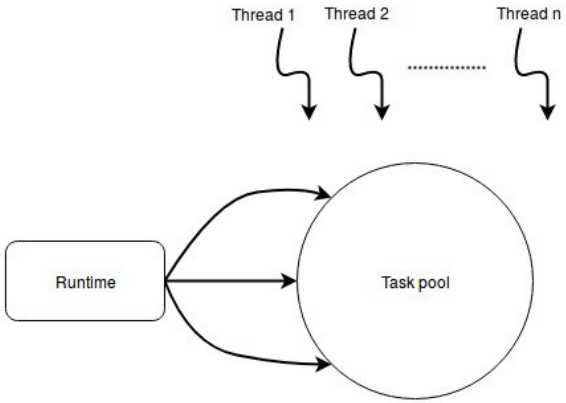
```
ddss_dgemm( ... ){
  ...
  for ( mi = 0; mi < mt; mi++ ){
    for ( ni = 0; ni < nt; ni++ ){
      for ( ki = 0; ki < kt; ki++ ){
        #pragma oss task in( TILE_A[mi][ki] ) \
        in( TILE_B[ki][ni] ) \
        inout( TILE_C[mi][ni] ) \
        shared( TILE_A, TILE_B, TILE_C ) \
        firstprivate( mi, ni, ki, betat )
        cblas_dgemm( CblasRowMajor,
          ( CBLAS_TRANSPOSE ) TRANS_A,
          ( CBLAS_TRANSPOSE ) TRANS_B,
          tile_size_m,
          tile_size_n,
          tile_size_k,
          ALPHA, TILE_A[mi][ki], tile_size_k,
                 TILE_B[ki][ni], tile_size_n,
          betat, TILE_C[mi][ni], tile_size_n );
      }
    }
  }
...
}
```

OpenMP vs OmpSs

```
#pragma omp parallel
#pragma omp master
{
    #pragma omp task depend(in:A[0:lda*ak]) \
                     depend(in:B[0:ldb*bk]) \
                     depend(inout:C[0:ldc*n])
    {
        core_dgemm(transa, transb,
                   m, n, k,
                   alpha, A, lda,
                          B, ldb,
                   beta,  C, ldc);
    }
}
```

```
#pragma oss task in(A[0:lda*ak]) \
                 in(B[0:ldb*bk]) \
                 inout(C[0:ldc*n])
{
        core_dgemm(transa, transb,
                   m, n, k,
                   alpha, A, lda,
                          B, ldb,
                   beta,  C, ldc);

}
#pragma oss taskwait
```





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

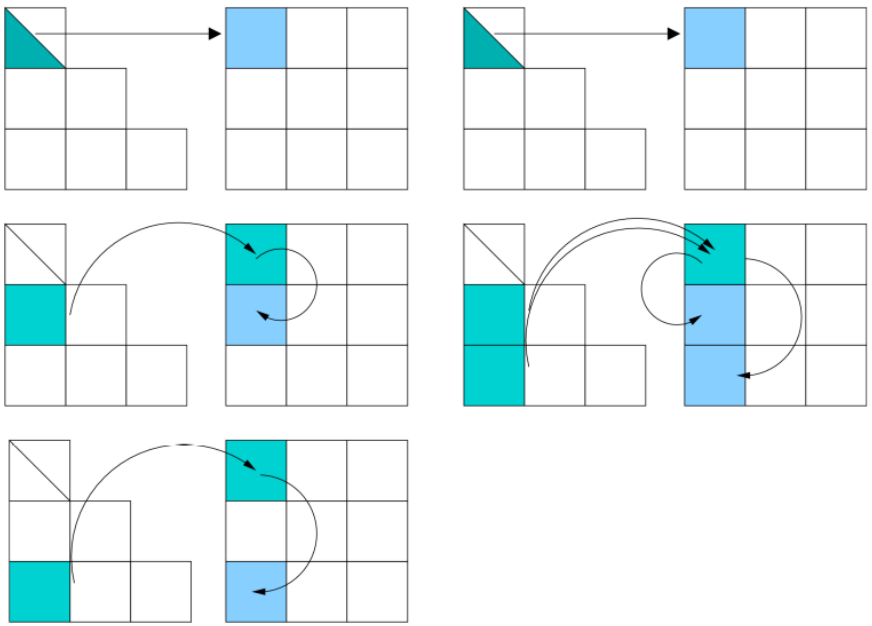
EXCELENCIA SEVERO OCHOA

1) Introduction
2) **TRSM in LASs**
3) NPGETRF in LASs
4) sLASs on small mat.
5) SpMV
6) Ongoing work

BLAS-3 Optimization

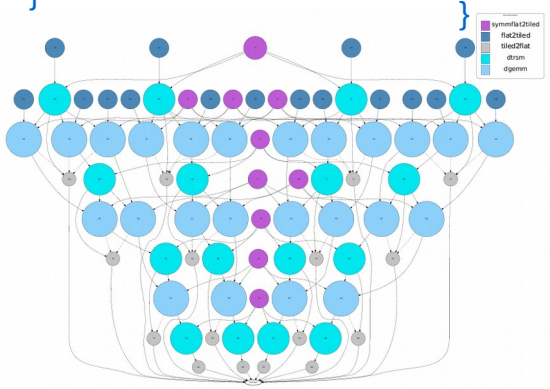
dtrsm (triagular solve)

- Join in one tasks multiple *dgemm that* compute one column, instead of one task per tile
  - Less number of tasks/dependences
  - Smaller DAG
  - Better exploitation of memory hierarchy
  - Tunable (Auto-tuning)
  - **Use of OmpSs regions**
  - Tile Size → size of the matrix and #cores
    - $36864^2 \rightarrow 768^2$



```
for ( d = 0; d < dt; d++){
   for ( c = 0; c < ct; c++){
         #pragma oss task ...
         dtrsm( … );
   }
   for ( c = 0; c < ct; c++) {
      for ( r = d; r < rt; r++)  {
         #pragma oss task …
         dgemm( … );
      }
   }
}
```

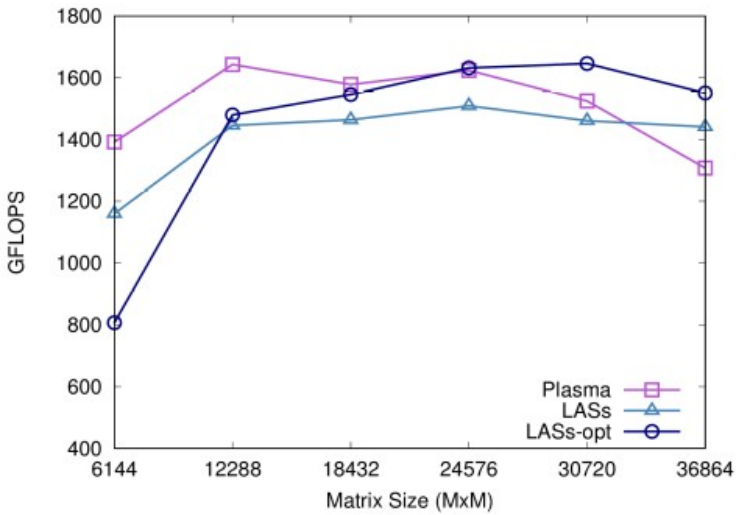
```
for ( d = 0; d < dt; d++){
   for ( c = 0; c < ct; c++){
      #pragma oss task …
      dtrsm( … );
   }
   for ( c = 0; c < ct; c++) {
      #pragma oss task \
        inout (TILEB[d:rt-1][c]) ...
      for ( r = d; r < rt; r++) {
         dgemm( … );
      }
   }
}
```

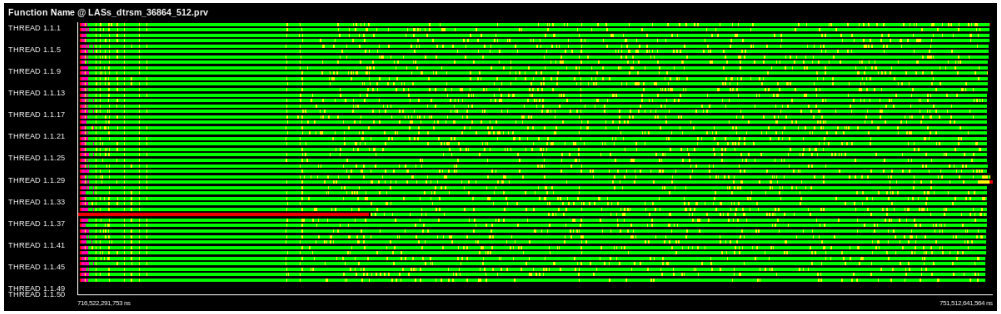


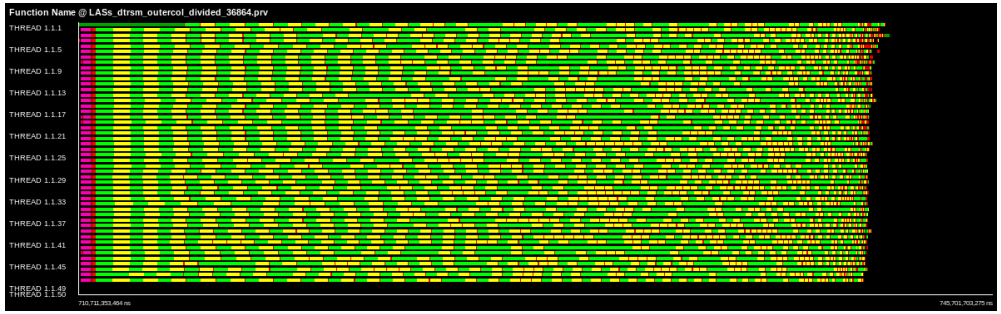
Performance Evaluation (Extrae + Paraver)

dtrsm

- LASs-opt
  - Faster than LASs
    - from 18432^2 in dtrsm and from 24576^2 in dtrmm
    - 5% (18432^2) → 12% (36864^2)
  - Faster than Plasma (OpenMP)
    - from 30720^2
    - 12 % (30720^2) → 15% (36864^2)
  - On small matrices the optimization is not effective
    - 6144^2 → tile_size = 128
    - Small tiles → small IPC (GLFOPS) in dgemm







Time(s)	6144	12800	18432	24576	30720	36864
Plasma	**0,16**	**1,12**	**3,96**	**9,20**	19,89	38,29
LASs	0.20	1,45	4,28	9,83	19,83	36,61
LASs_opt	0,29	1,50	*4,06*	*9,44*	**17,45**	**32,29**

sLASs

dtrsm (triagular solve)

- Depending on the matrix size and number of cores, the optimization is effective or not
  - We need a mechanism that allows us to adapt the execution, depending on:
    - the size of the matrix
    - the cores available
  - Minimum modifications on the code
    - Low maintainability cost
    - Portability and adaptability
- This is implemented by using:
  - Regions
  - Weak dependences
  - Final clause

```
for ( d = 0; d < dt; d++){
    for ( c = 0; c < ct; c++){
        #pragma oss task ...
        dtrsm( … );
    }
    for ( c = 0; c < ct; c++) {
        for ( r = d; r < rt; r++)  {
            #pragma oss task …
            dgemm( … );
        }
    }
}
```

+

```
for ( d = 0; d < dt; d++){
    for ( c = 0; c < ct; c++){
        #pragma oss task …
        dtrsm( … );
    }
    for ( c = 0; c < ct; c++) {
        #pragma oss task \
         inout (TILEB[d:rt-1][c]) ...
        for ( r = d; r < rt; r++) {
            dgemm( … );
        }
    }
}
```

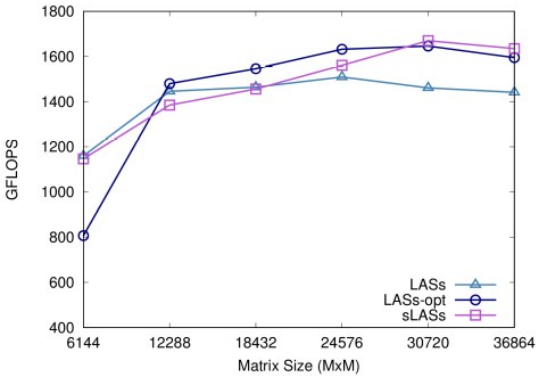
=

```
smart_dtrsm(is_final);

for ( d = 0; d < dt; d++){
    for ( c = 0; c < ct; c++){
        #pragma oss task …
        dtrsm( … );
    }
    for ( c = 0; c < ct; c++) {
        #pragma oss task \
         weakinout (TILEB[d:rt-1][c]) …
         final(is_final)
        for ( r = d; r < rt; r++) {
          #pragma oss task \
           inout(TILEB[r][c]) ...
          dgemm( … );
        }
    }
}
```

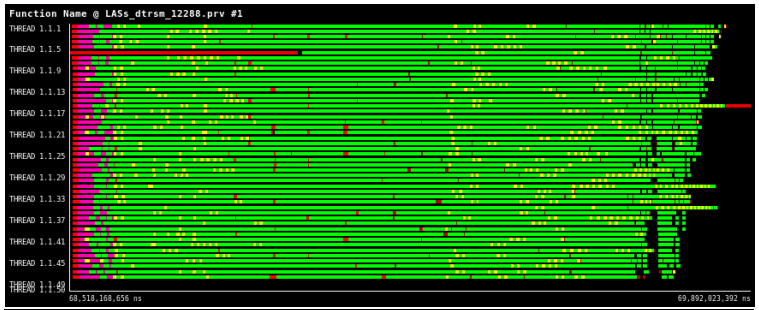
Performance Evaluation (Extrae + Paraver)

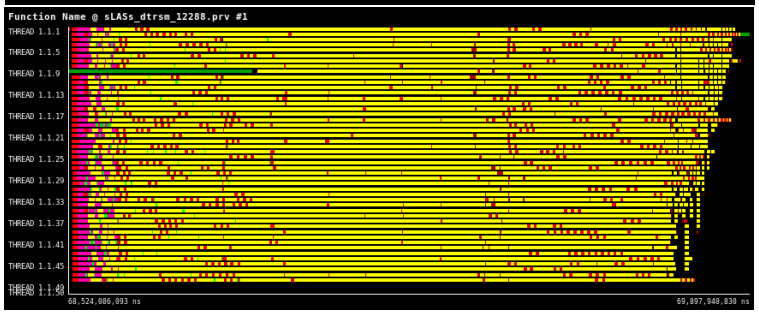
dtrsm@sLASs

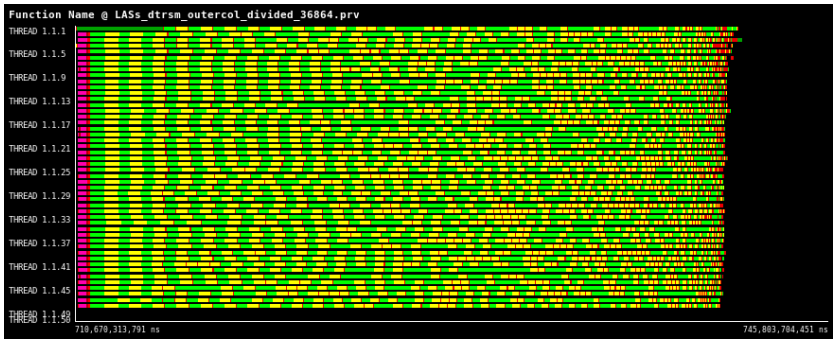
- Obtains similar results to the reference implementations
  - Nesting (condition of the final OmpSs clause = false):
    - from 6144^2 to 18432^2
      - Insufficient parallelism to do join on dgemm
      - 6144 / 512 (Default tile size) < number of cores (48)
    - Instantiate more tasks
      - 0,82% (18432^2)
  - No nesting (condition of the final OmpSs clause = true)
    - From 24576^2 to 36864^2
      - Big matrices → join on dgemm and bigger tiles size
    - No overhead

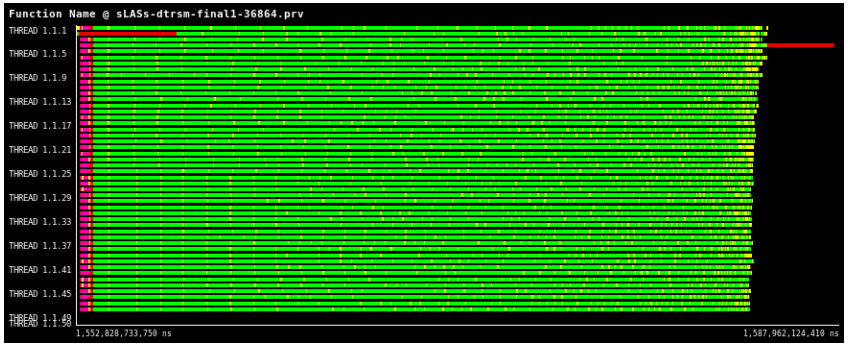


Time(s)	6144	12288	18432	24576	30720	36864
LASs	**0,20**	**1,45**	4,28	9,83	19,83	36,61
LASs_opt	0,29	1,50	**4,06**	**9,44**	17,45	32,29
sLASs	**0,20**	1,54	4,35	9,51	**17,36**	**30,63**









Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

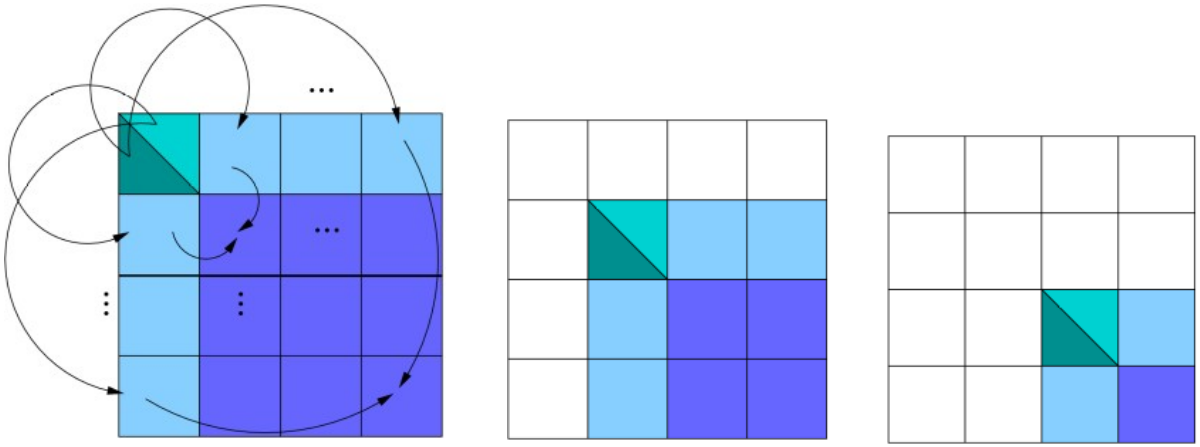
EXCELENCIA SEVERO OCHOA

1) Introduction
2) TRSM in LASs
3) **NPGETRF in LASs**
4) sLASs on small mat.
5) SpMV
6) Ongoing work

sLASs

dnpgetrf (LU factorization without pivoting)

- Np → no pivoting
  - Well-conditioned matrices
  - For the sake of performance evaluation
    - To compare with pivoting versions
- Dynamic behavior
  - The parallelism (# dgemm) is reduced along the factorization
    - In the first steps → the optimization can be effectively exploited
    - In the last steps → the optimization is not effective
  - The "is_final" condition must be computed every step
    - smart_dnpgetrf

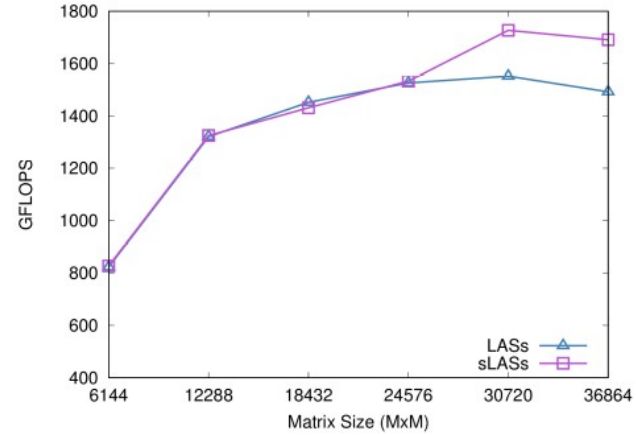


```
for ( d = 0; d < dt; d++){
    #pragma oss task …
    dnpgetrf( … );
    for ( r = d+1; r < rt; r++){
        #pragma oss task …
        dtrsm( … );
    }
    for ( c = d+1; c < ct; c++){
        #pragma oss task …
        dtrsm( … );
    }
    for ( c = d+1; c < ct; c++){

        smart_dnpgetrf(is_final);

        #pragma oss task \
        weakin(TILEA[d+1:rt-1][d]) …
        final(is_final)
        for ( r = d+1; r < rt; r++) {
            #pragma oss task \
             in(TILEA[r][d]) …
            dgemm( … );
        }
    }
}
```

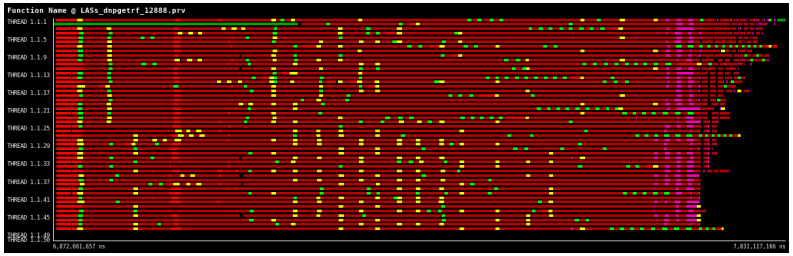
Performance Evaluation (Extrae + Paraver)

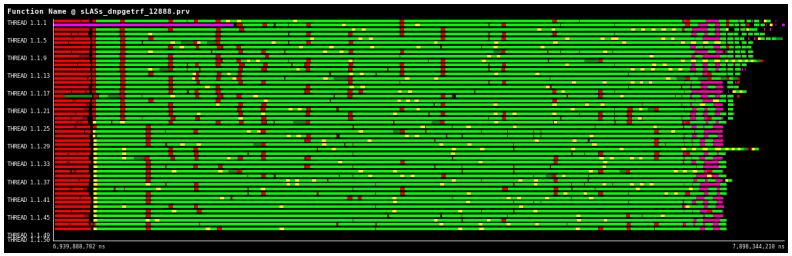
dnpgetrf@sLASs

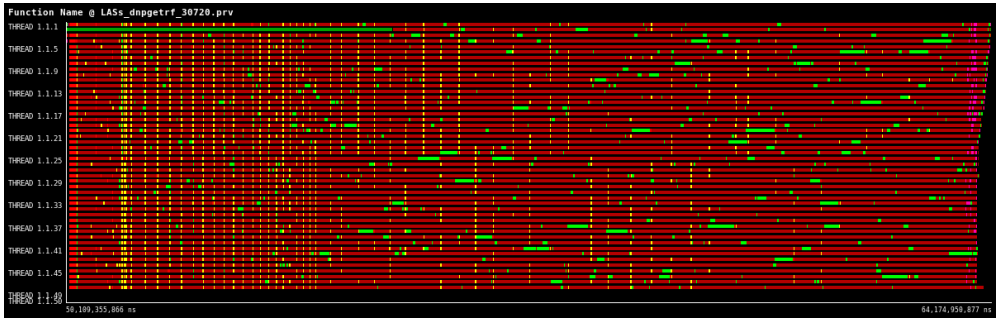
- Nesting (condition of the final OmpSs clause = false):
  - from 6144^2 to 18432^2
    - Insufficient parallelism to do join on dgemm
    - 6144 / 512 (Default tile size) < number of cores (48)
  - Instantiate more tasks
    - 0,79% (12288^2)
- No nesting (condition of the final OmpSs clause = true)
  - From 24576^2 to 36864^2
    - **In the first steps** → high parallelism
    - Big matrices → join on dgemm and bigger tiles size
  - Time Reduction in the tasks instantiation
    - 36,03% → 17,31%
  - Increase of IPC in dgemm
    - ~1,7 → ~2,25

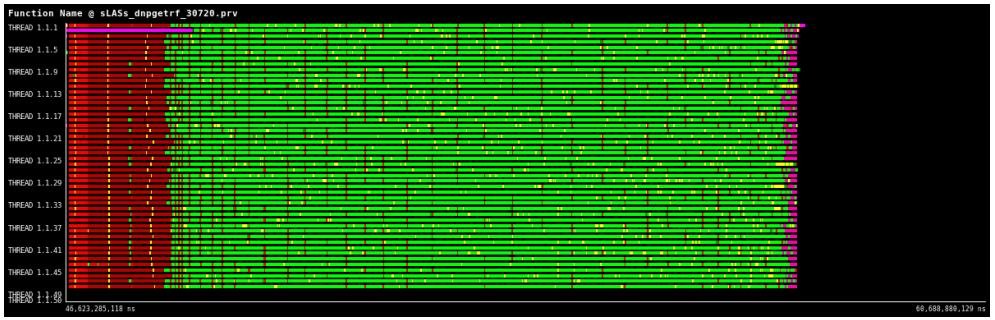


Time(s)	6144	12288	18432	24576	30720	36864
LASs	**0.19**	**0,93**	**2,90**	**6,63**	13,10	22,40
sLASs	**0,19**	**0,93**	2,96	7,04	**11,45**	**20,28**









Barcelona Supercomputing Center
Centro Nacional de Supercomputación

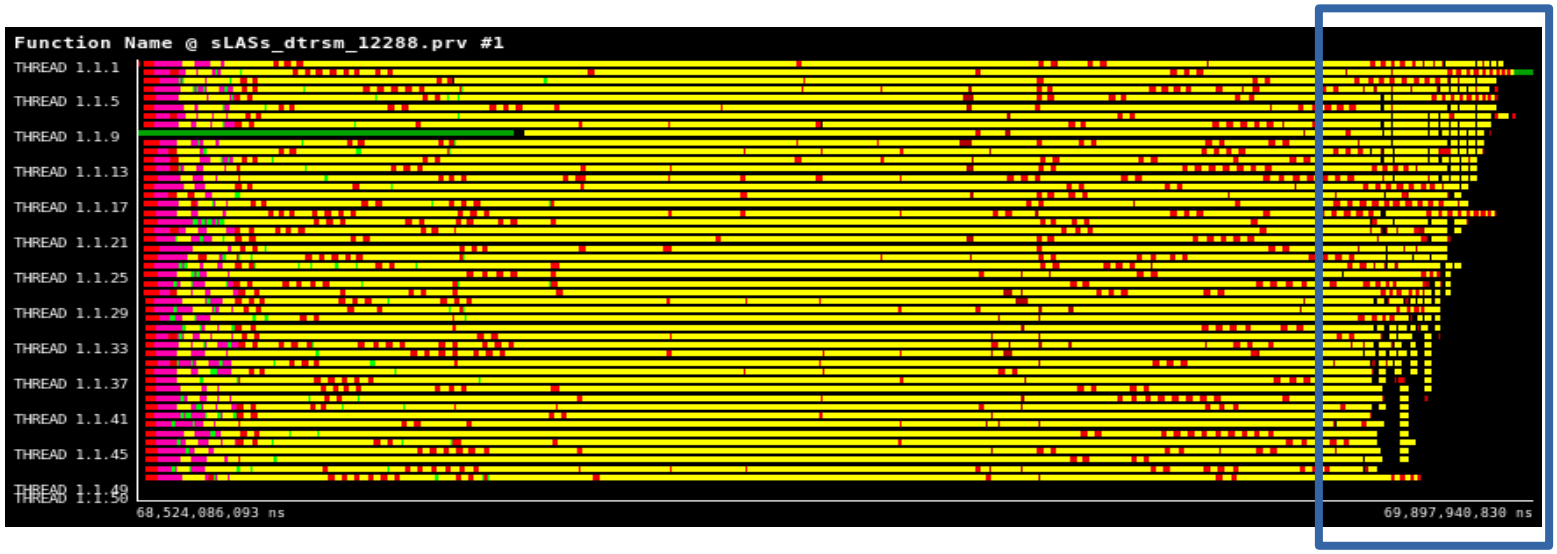
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

1) Introduction
2) TRSM in LASs
3) NPGETRF in LASs
4) **sLASs on small mat.**
5) SpMV
6) Ongoing work

SLASs – Improving small matrices performance

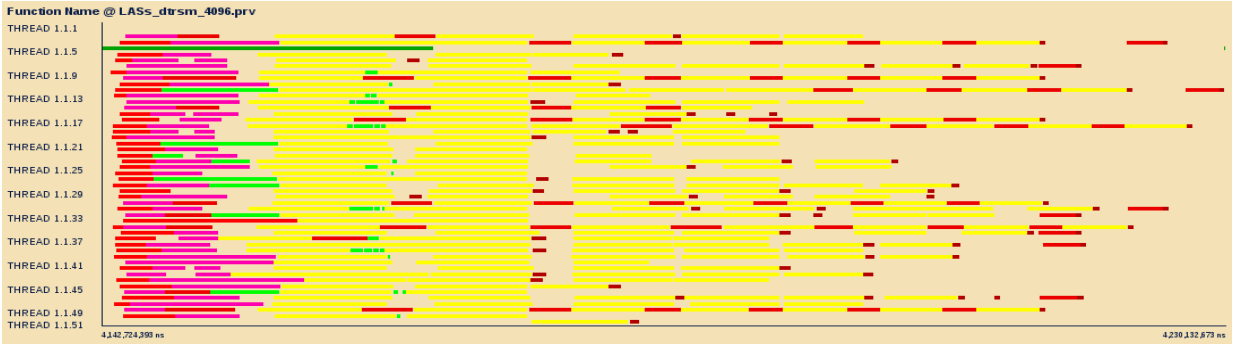
Why small matrices?



- SLASs
  - N=12288, b=512, 48 cores
- There exists a significant workload imbalance (20% time)
- Improving workload balance benefits from small to large matrices.

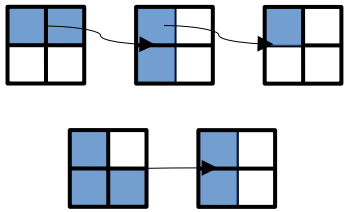
SLASs – Improving small matrices performance

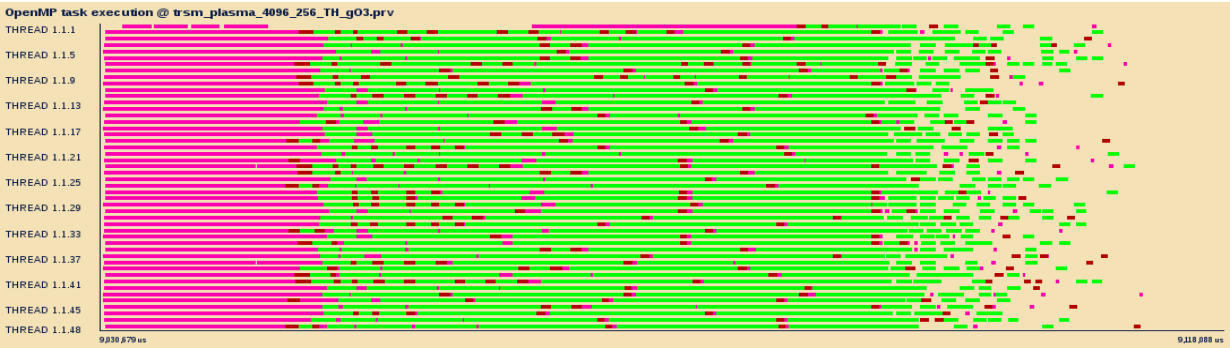
From LASs to sLASs



- SLASs
  - N=4096, b=512, 48 cores
- Big workload imbalance.

- SLASs + opt. gemm/trsm tasks
  - N=4096, b=512, 48 cores
- Workload imbalance is considerably reduced.





- PLASMA
  - N=4096, b=512, 48 cores
- Big workload imbalance.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

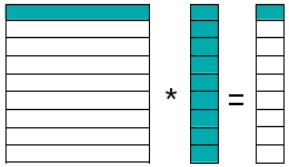
EXCELENCIA SEVERO OCHOA

1) Introduction
2) TRSM in LASs
3) NPGETRF in LASs
4) sLASs on small mat.
5) **SpMV**
6) Ongoing work

LASs

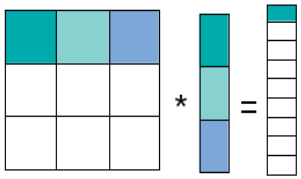
dspmv (sparse matrix-vector multiplication) Y= ALPHA * A * X + Y * BETA

1 Task per Row



- High number of tasks
- Tasks are not big enough

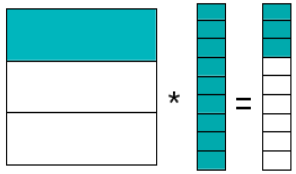
Blocking



- Requires preprocessing
- Data dependences

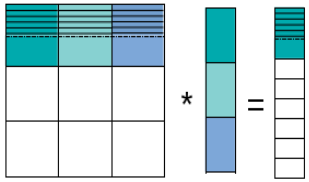
in-2004

Taskloop



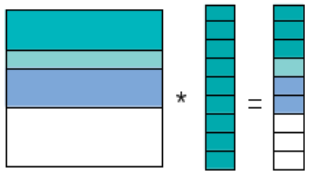
- Grainsize needs to be
- determined
- Easy to implement

Blocking + Nesting



- Requires preprocessing
- Data dependences

Grouping



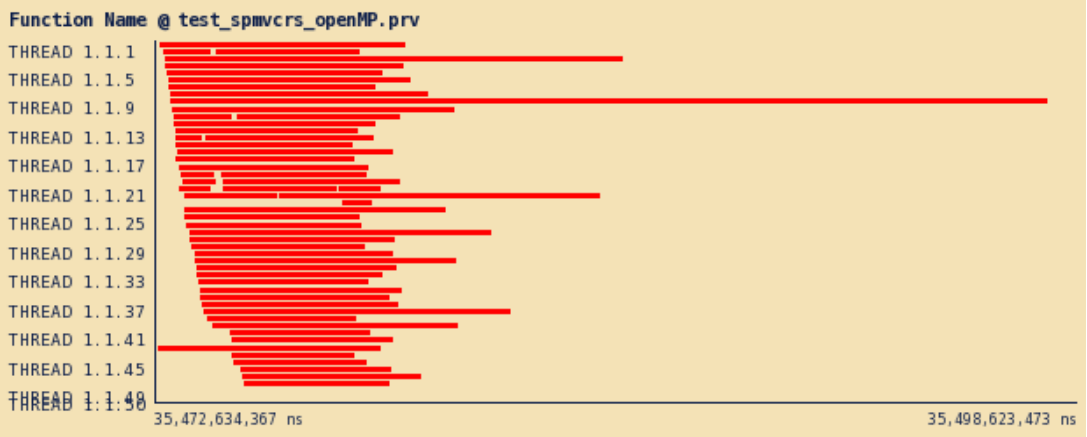
- Requires extra calculations to create groups

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

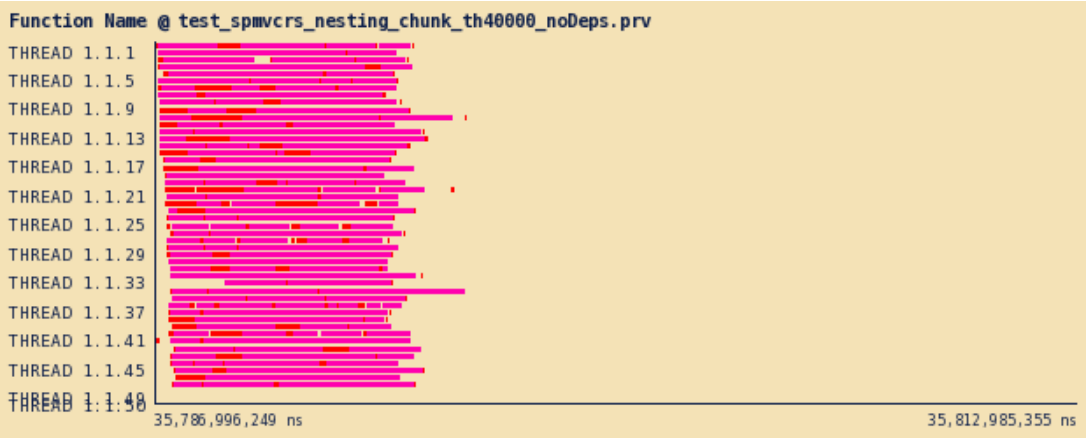
Performance Evaluation (Extrae + Paraver)

Optimization on dspmv taskloop using OmpSs tasks

```
#pragma oss taskloop grainsize ...
for ( r = 0; r < nRows; r++){
    Sval = 0.0;
    for ( c = 0; c < nCols; c++) {
        val = VAL_A[ROW_A[r]+c];
            col = COL_A[ROW_A[r]+c];
            sval += val* X[col]*ALPHA;
    }
    Y[r] = sval + Y[r] *BETA;
}
```




```
for ( rb = 0; rb < nRows; rb++)
{
  #pragma oss task
  {
   nnz = number_of_non_zeros_in_row_block(rb);
   #pragma omp taskloop num_tasks(nnz/th) if (nnz > th)
   for ( r = rb; r <r + rb ; r++){
       Sval = 0.0;
       for ( c = 0; c < nCols; c++) {
               val = VAL_A[ROW_A[r]+c];
               col = COL_A[ROW_A[r]+c];
               sval += val* X[col]*ALPHA;
       }
       Y[r] = sval + Y[r] *BETA;
   }
  }
}
```



Note that ***th*** is a threshold set to 40,000, a number determined experimentally.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

- Intel Xeon Platinium 8160 (MareNostrum) 2x 24 cores

Ongoing work

- Iterative sparse solvers
  - CG
- Direct sparse solvers
- Improving GESV performance
  - Small matrices optimizations

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

THANK YOU!

FOR MORE INFO:
pm.bsc.es/mathlibs/lass

www.bsc.es