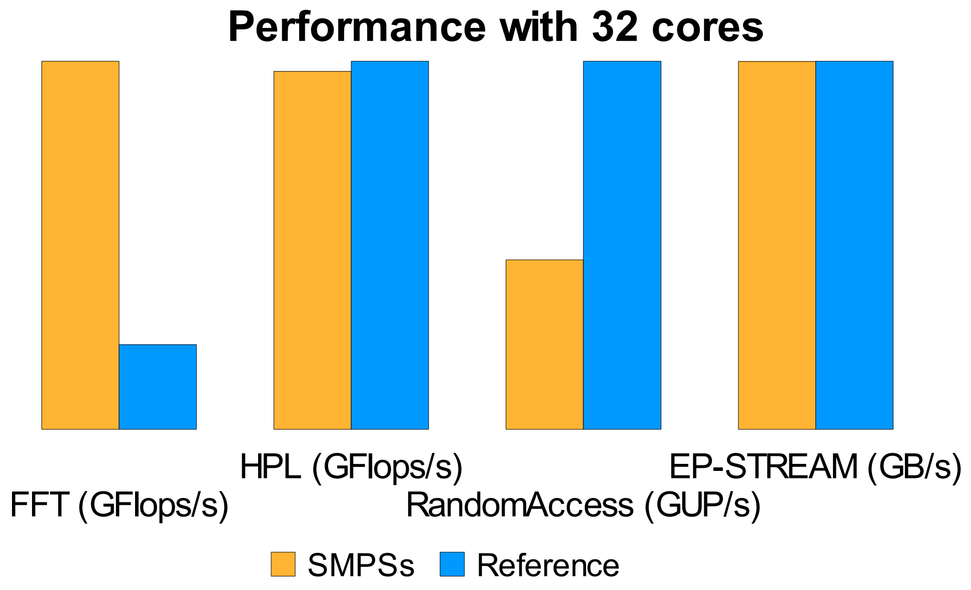
Departament d'Arquitectura de Computadors
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

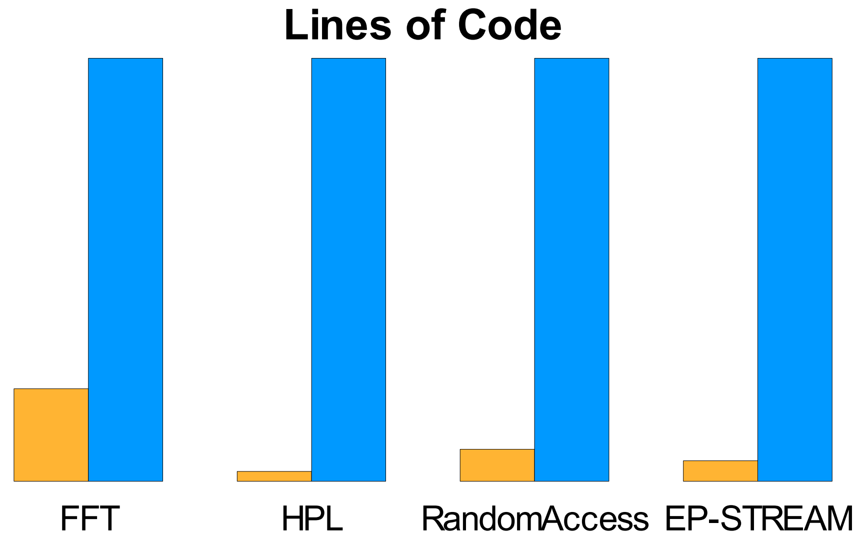
SMPSs Submission to HPCC 2010 Class 2 Competition

Josep M. Perez, Rosa M. Badia, Jesus Labarta, Eduard Ayguade

Index



Performance with 32 cores
FFT (GFlops/s)
HPL (GFlops/s)
RandomAccess (GUP/s)
EP-STREAM (GB/s)
SMPSs
Reference



Lines of Code
FFT
HPL
RandomAccess
EP-STREAM

Programming Model and Runtime

- StarSs Model
  - Sequential single address space program
  - Task-based
  - Pragmas: specify tasks and directionality of arguments
- Runtime
  - Compute dependencies at runtime
  - Data access (locality) information
  - Scheduling

```
#pragma css task                    \
      input (<params>)              \
      output (<params>)             \
      inout (<params>)              \
      reduction (<params>)          \
      [highpriority]
<function definition | function declaration>
```

```
#pragma css barrier
```

```
#pragma css wait on(<vars>)
```

- SMPSs runtime with region support
  - Strided and aliased arguments
  - Locality-aware scheduler

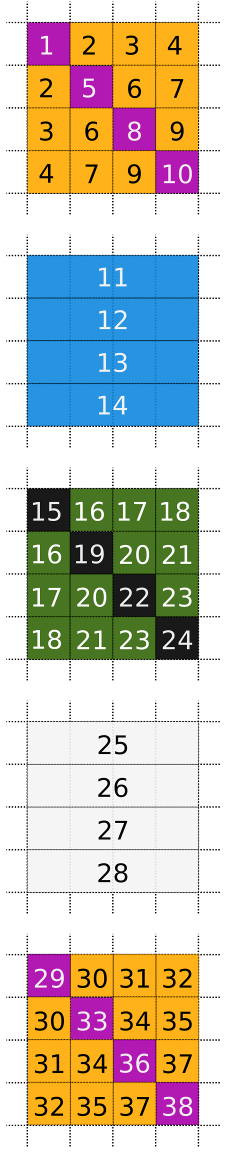
Target Platform & Measurement

- SGI Altix 4700
  - Highly NUMA shared memory machine
  - 32 memory nodes
  - 2 dual-core Itanium (Montecito) per node
  - 1.6 GHz, 8 MB shared L3 cache
  - 32 GB 533 MHz memory per node
- Libraries
  - GotoBLAS 1.21
  - FFTW 3.2.2
- Compiler
  - ICC 11.0

Sequential FFT

Main Code

```
void fft (double _Complex A[N2][N2]) {
    // 1. Transpose
    for (long i=0; i<N2; i+=TR_BS) {
        trsp_blk (&A[i][i]);
        for (long j=i+TR_BS; j<N2; j+=TR_BS)
            trsp_swap (&A[i][j], &A[j][i]);
    }

    // 2. First FFT round
    for (long j=0; j<N2; j+=FFT_BS)
        fft1d(&A[j][0]);

    // 3 & 4. Twiddle and Transpose
    for (long i=0; i<N2; i+=TR_BS) {
        tw_trsp_blk (i, &A[i][i]);
        for (long j=i+TR_BS; j<N2; j+=TR_BS)
            tw_trsp_swap (i, j, &A[i][j], &A[j][i]);
    }

    // 5. Second FFT round
    for (long j=0; j<N2; j+=FFT_BS)
        fft1d(&A[j][0]);

    // 6. Transpose
    for (long i=0; i<N2; i+=TR_BS) {
        trsp_blk (&A[i][i]);
        for (long j=i+TR_BS; j<N2; j+=TR_BS)
            trsp_swap (&A[i][j], &A[j][i]);
    }
}
```



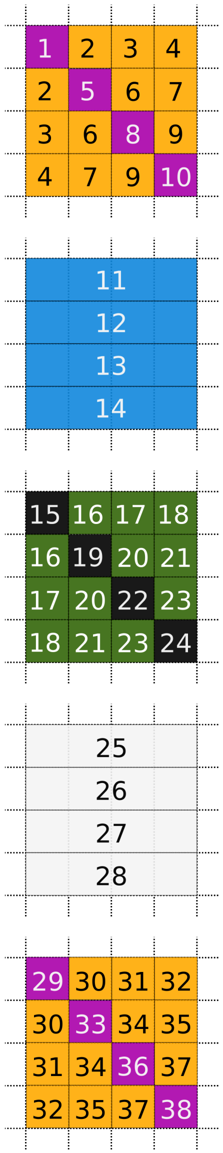
Functions

```
void trsp_blk(double _Complex blk[N2][N2]);

void trsp_swap (
        double _Complex blk1[N2][N2],
        double _Complex blk2[N2][N2]);

void fft1d (double _Complex panel[FFT_BS][N2]);

void tw_trsp_blk(long I, double _Complex panel[N2][N2]);

void tw_trsp_swap (long I, long J,
        double _Complex blk1[N2][N2],
        double _Complex blk2[N2][N2]);
```

Parallel FFT in SMPSs

Main Code

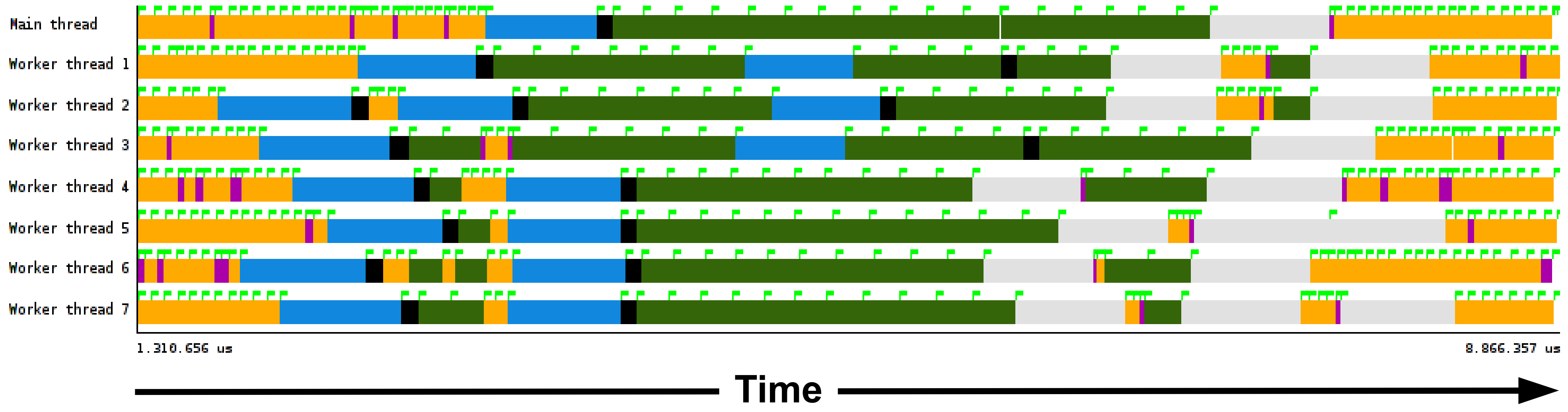
```
void fft (double _Complex A[N2][N2]) {
   // 1. Transpose
   for (long i=0; i<N2; i+=TR_BS) {
       trsp_blk (&A[i][i]);
       for (long j=i+TR_BS; j<N2; j+=TR_BS)
           trsp_swap (&A[i][j], &A[j][i]);
   }

   // 2. First FFT round
   for (long j=0; j<N2; j+=FFT_BS)
       fft1d(&A[j][0]);

   // 3 & 4. Twiddle and Transpose
   for (long i=0; i<N2; i+=TR_BS) {
       tw_trsp_blk (i, &A[i][i]);
       for (long j=i+TR_BS; j<N2; j+=TR_BS)
           tw_trsp_swap (i, j, &A[i][j], &A[j][i]);
   }

   // 5. Second FFT round
   for (long j=0; j<N2; j+=FFT_BS)
       fft1d(&A[j][0]);

   // 6. Transpose
   for (long i=0; i<N2; i+=TR_BS) {
       trsp_blk (&A[i][i]);
       for (long j=i+TR_BS; j<N2; j+=TR_BS)
           trsp_swap (&A[i][j], &A[j][i]);
   }
}
```

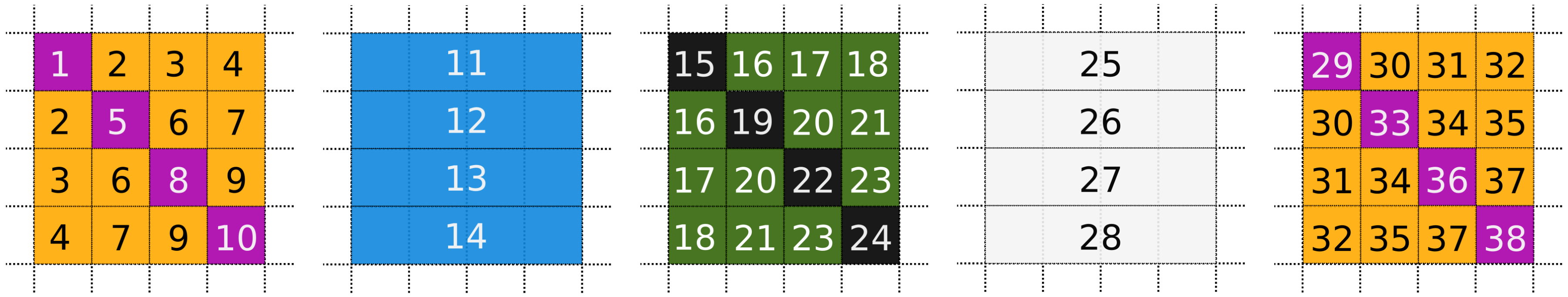


Tasks

```
#pragma css task inout(blk{0:TR_BS}{0:TR_BS})
void trsp_blk(double _Complex blk[N2][N2]);

#pragma css task inout \
        (blk1{0:TR_BS}{0:TR_BS}, \
        blk2{0:TR_BS}{0:TR_BS})
void trsp_swap (
        double _Complex blk1[N2][N2],
        double _Complex blk2[N2][N2]);

#pragma css task inout(panel)
void fft1d (double _Complex panel[FFT_BS][N2]);

#pragma css task input(I) inout(panel{0:TR_BS}{0:TR_BS})
void tw_trsp_blk(long I, double _Complex panel[N2][N2]);

#pragma css task input (I, J) inout \
        (blk1{0:TR_BS}{0:TR_BS}, \
        blk2{0:TR_BS}{0:TR_BS})
void tw_trsp_swap (long I, long J,
        double _Complex blk1[N2][N2],
        double _Complex blk2[N2][N2]);
```

Global FFT in SMPSs



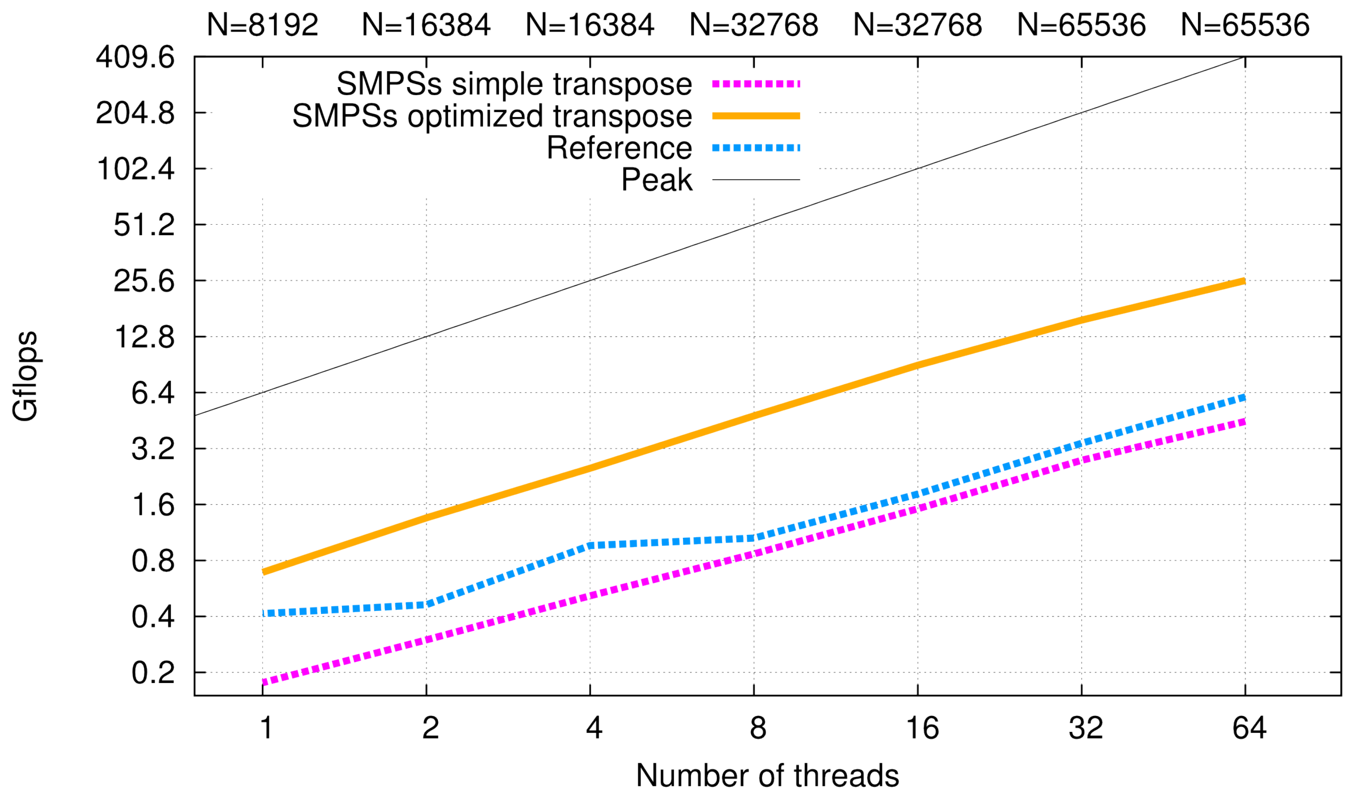


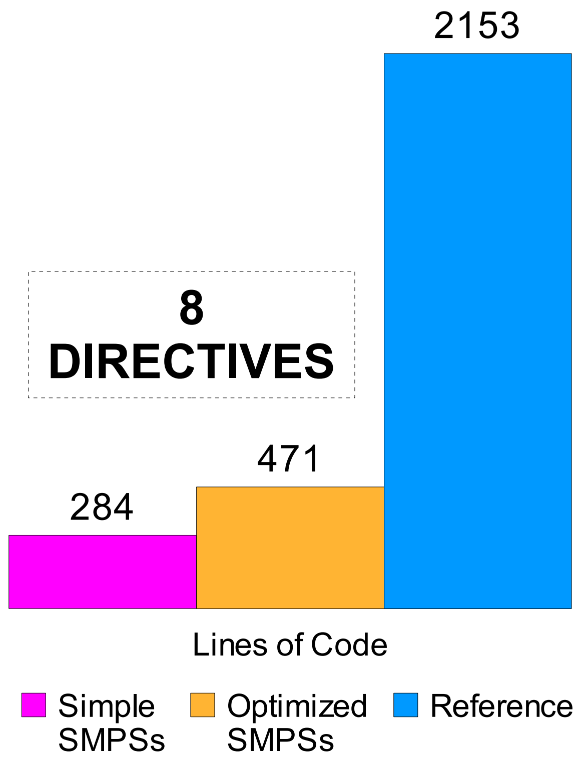
🟪 # trsp_blk	🟦 # fft1d (first round)	⬛ # tw_trsp_blk	⬜ # fft1d (second round)
🟧 # trsp_swap		🟩 # tw_trsp_swap	

Departament d'Arquitectura de Computadors
UNIVERSITAT POLITÈCNICA DE CATALUNYA
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Global FFT Performance and Code Size

- Two versions:
  - One with simple transposition tasks
  - One with optimized blocked transpositions

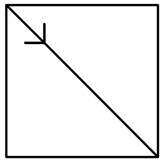


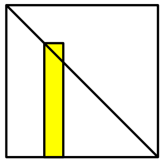


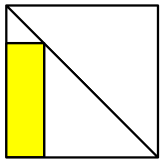
Sequential HPL

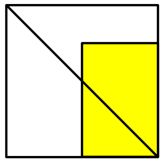
LAPACK LU Code (C)

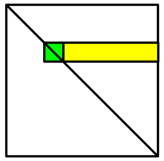
```
void tiled_dgetrf(integer NB, integer M, integer N,
   integer LDA, double A[N][LDA], integer IPIV[]) {

   integer jb = min(min(M, N), NB);

   if (M <= NB || N <= NB)
      dgetrf_tile(M, N, LDA, A, IPIV);

   else
      for (integer j=0; j < min(M, N); j += jb) {

         jb = min(min(M, N)-j, jb);
         dgetrf_tile(M-j, jb, LDA, &A[j][j], &IPIV[j]);

         if (j != 0) {
            tiled_add_scalar(jb, min(M-j, jb), j, &IPIV[j]);

            tiled_dlaswp(jb, M, j, LDA, A, j, j+jb-1, IPIV);
         }

         if (j+jb < N) {
            tiled_dlaswp(jb, M, N-j-jb, LDA, &A[j+jb][0], j, j+jb-1, IPIV);

            tiled_dtrsm(jb, jb, N-j-jb, 1.0, LDA, &A[j][j], LDA, &A[j+jb][j]);

            if (j+jb < M)
               tiled_dgemm(jb, M-j-jb, N-j-jb, jb, -1.0, LDA, &A[j][j+jb], LDA, &A[j+jb][j], 1.0, LDA, &A[j+jb][j+jb]);
         }
      }
}
```

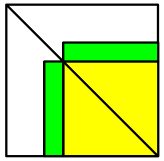




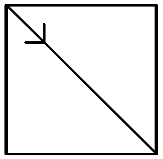


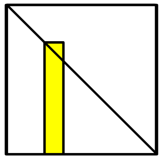






Sequential HPL

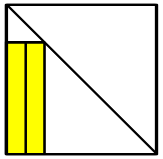




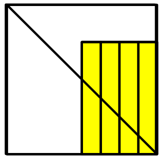
Function Decomposition

Tiles

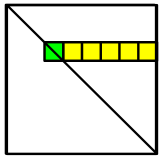
```
void tiled_add_scalar(...) {
    for (...)
        add_scalar_tile(...);
}
```



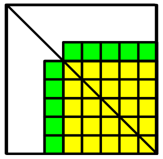
```
void tiled_dswap(...) {
    for (...)
        dswap_tile(...);
}
```



```
void dtrsm_tile(int M,..., double A[M][LDA], double B[N][LDB]) {
    dtrsm_("Left", ..., A, ..., B, ...);
}
```

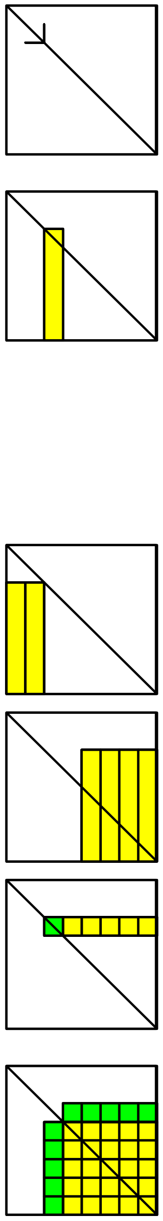


```
void tiled_dtrsm(integer NB, integer M, integer N, double
ALPHA, integer LDA, double const A[M][LDA], integer LDB,
double B[N][LDB]) {
    for (integer i = 0; i < N; i += NB)
        dtrsm_tile(M, min(NB, N-i), ALPHA, LDA, A, LDB, &B[i][0]);
}
```



```
void tiled_dgemm(...) {
    for (...)
        for (...)
            for(...)
                dgemm_tile(...);
}
```

Global HPL in SMPSs



Function Decomposition

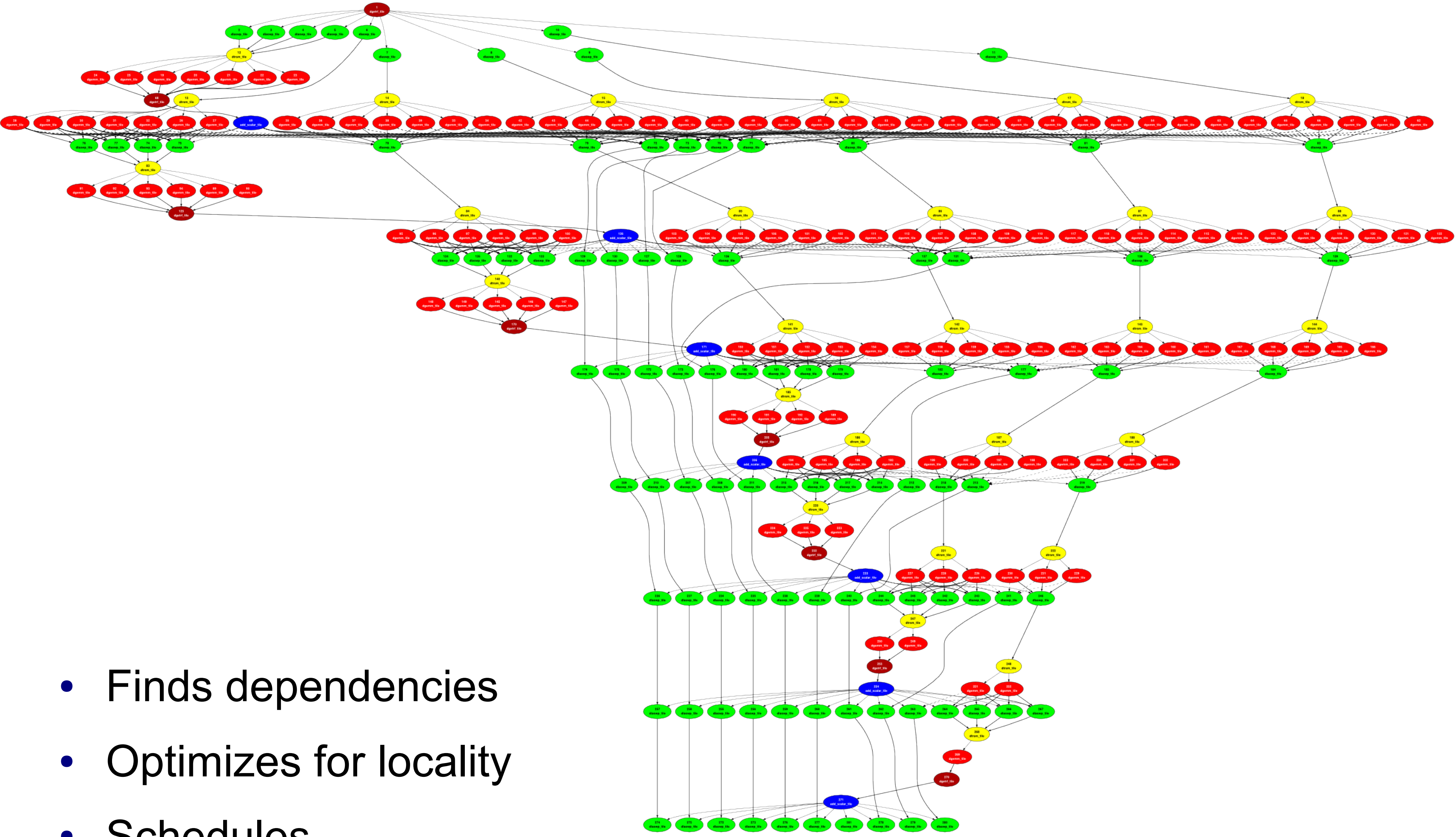
```
void tiled_add_scalar(...) {
    for (...)
        add_scalar_tile(...);
}

void tiled_dswap(...) {
    for (...)
        dswap_tile(...);
}

void tiled_dtrsm(integer NB, integer M, integer N, double
ALPHA, integer LDA, double const A[M][LDA], integer LDB,
double B[N][LDB]) {
    for (integer i = 0; i < N; i += NB)
        dtrsm_tile(M, min(NB, N-i), ALPHA, LDA, A, LDB, &B[i][0]);
}

void tiled_dgemm(...) {
    for (...)
        for (...)
            for(...)
                dgemm_tile(...);
}
```

Tasks

```
#pragma css task        input( M, N, ..., A{0:M}{0:M} ) \
                        inout( B{0:N}{0:M} )
void dtrsm_tile(int M,..., double A[M][LDA], double B[N][LDB]) {
    dtrsm_("Left", ..., A, ..., B, ...);
}
```

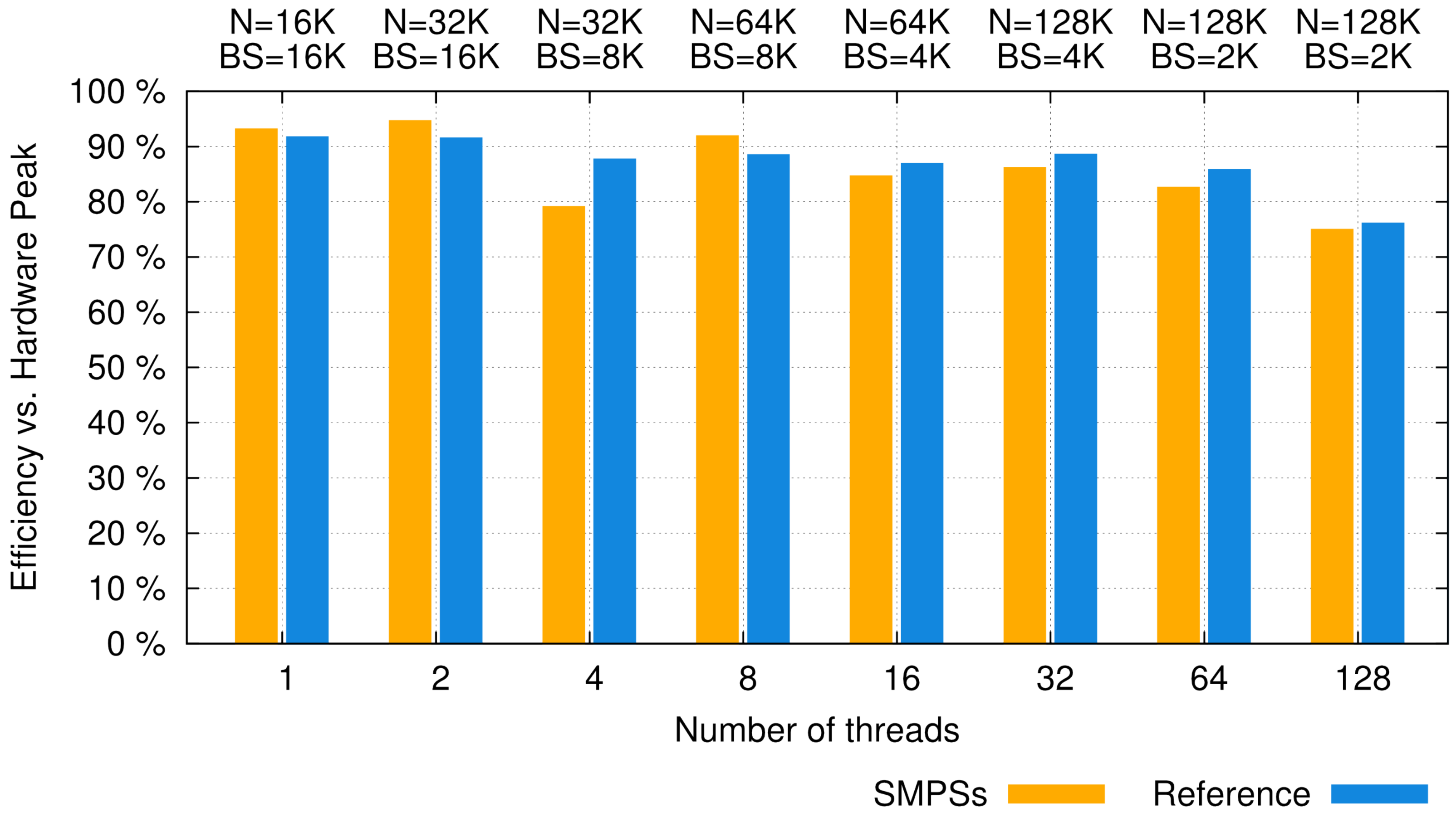
🟩 input 🟨 inout

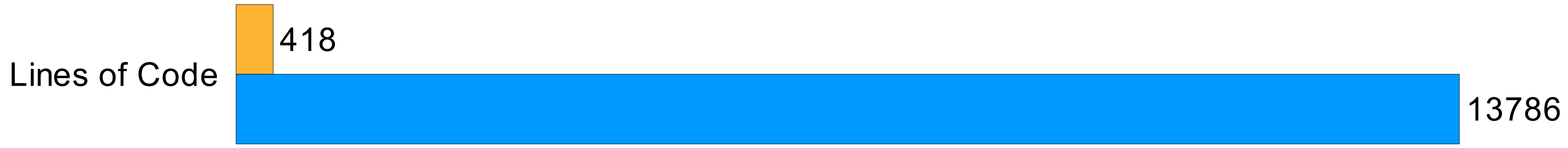
Runtime



- Finds dependencies
- Optimizes for locality
- Schedules

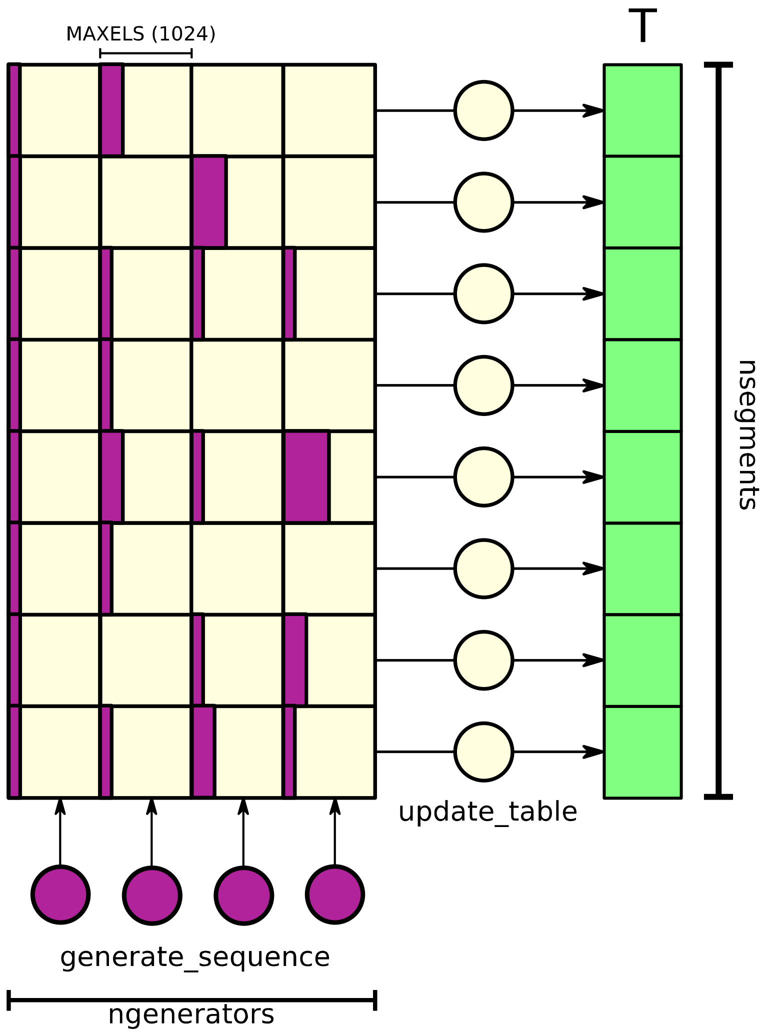
Global HPL Performance and Code Size





Departament d'Arquitectura de Computadors
UNIVERSITAT POLITÈCNICA DE CATALUNYA
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Global Random Access

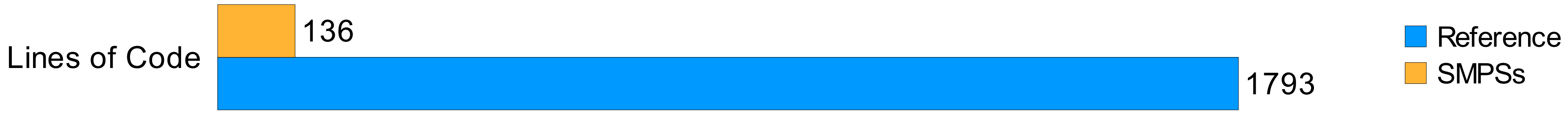


```
for (long total = 0; total < NUPDATES; total += MAXELS*ngenerators) {
        for (int gen=0; gen < ngenerators; gen++)
                generate_sequence(&seq[gen][0][0], &fills[gen][0], &lrand[gen]);

        for (int seg = 0; seg < NSEGS; seg++)
                update_table(ngenerators, seg, seq, fills, &T[seg], seg*SEGSIZE);
}
```

```
#pragma css task output(seq, fills)  inout(lrand) highpriority
void generate_sequence(uint64_t seq[NSEGS][MAXELS],
         int fills[NSEGS], uint64_t *lrand)

#pragma css task input(seg, seq, fills, offset) inout(T)
void update_table(int seg, uint64_t seq[ngenerators][NSEGS][MAXELS],
         int fills[NSEGS][NSEGS], uint64_t T[SEGSIZE], uint64_t offset)
```

Global performance (32 threads, 32 GB):

SMPSs	Reference
0.01450 GU/s	0.03150 GU/s



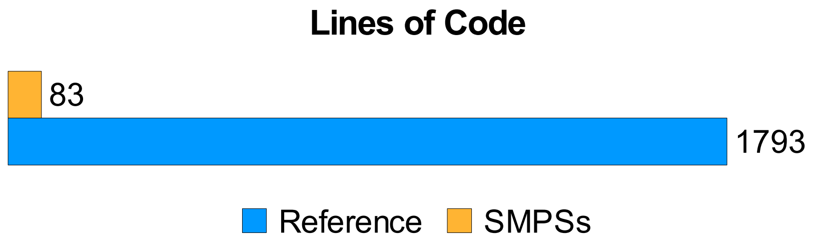
Global EP-Stream Triad

```
#pragma css task input(size, b, c, scalar) output(a)
void triad(long size, double a[size], double b[size], double c[size], double scalar) {
        for (long i = 0; i < size; i++)
                a[i] = b[i] + scalar*c[i];
}

for (long i = 0; i < N; i+= BSIZE)
    initialize_segment(BSIZE, &a[i], &b[i], &c[i], 0, 2, 1);

#pragma css barrier
for (int rep = 0; rep < NTIMES; rep++) {
    START_TIME();
    for (long i = 0; i < N; i += BSIZE)
        triad(BSIZE, &a[i], &b[i], &c[i], scalar);
    #pragma css barrier
    STOP_TIME();
    total_time[rep] = GET_TIME();
}
```

- Implementation:
  - Divide the vectors one piece per thread
  - One task instance per thread
  - Locality

Per thread performance (32 threads, 96 GB):

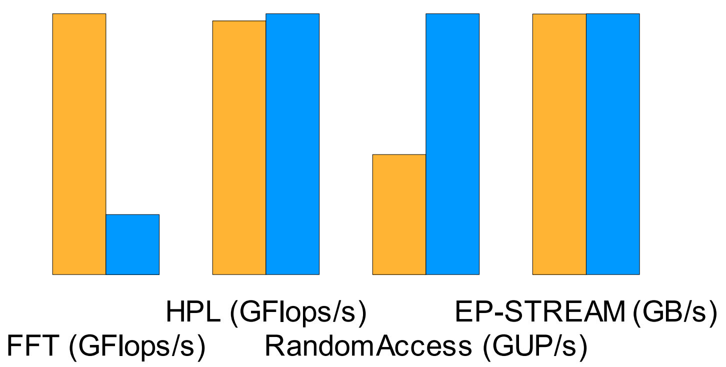
SMPSs	Reference	
	Original (t_{min})	Modified (t_{avg})
41.14 GB/s	43.15 GB/s	41.19 GB/s



Conclusions

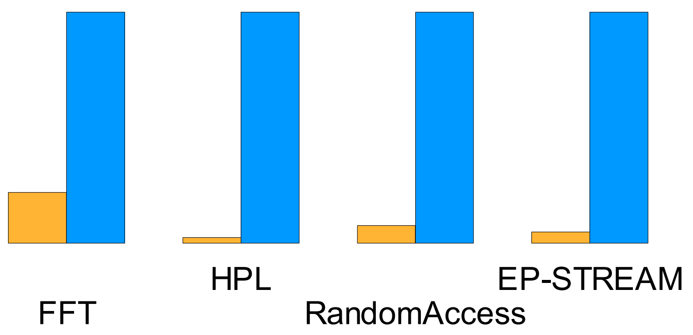
	FFT	HPL	RandomAccess	EP-STREAM
Incremental parallelization	X	X	X	X
Locality exploitation	X	X	X	X
Asynchrony and overlap	X	X		
Reuse of existing binaries	X	X		
Reuse of existing codes		X		

Performance with 32 cores



🟧 SMPSs
🟦 Reference

Lines of Code



http://www.bsc.es/smpsuperscalar

Departament d'Arquitectura de Computadors
UNIVERSITAT POLITÈCNICA DE CATALUNYA
Barcelona Supercomputing Center
Centro Nacional de Supercomputación