精密PA可視化機能の利用

Tuning

- List of Messages by Compiler (Compile List)
- 精密PA可視化機能 (Excel)
  (Precision PA Visibility Function)

List of Messages by Compiler (Compile List)

```
F90          = mpifrtpx
F90OPTFLAGS= -Kfast,openmp -Qt
F90FLAGS     = $(F90OPTFLAGS)
```

- -Qt
  - List of Messages by Compiler (Compile List)
  - *.lst
  - Fortran Only
- In C, “-Qt” is not avilable
  - Please use “-Nsrc”
  - Displayed on screen

Current version of C/C++ compiler can produce list of messages

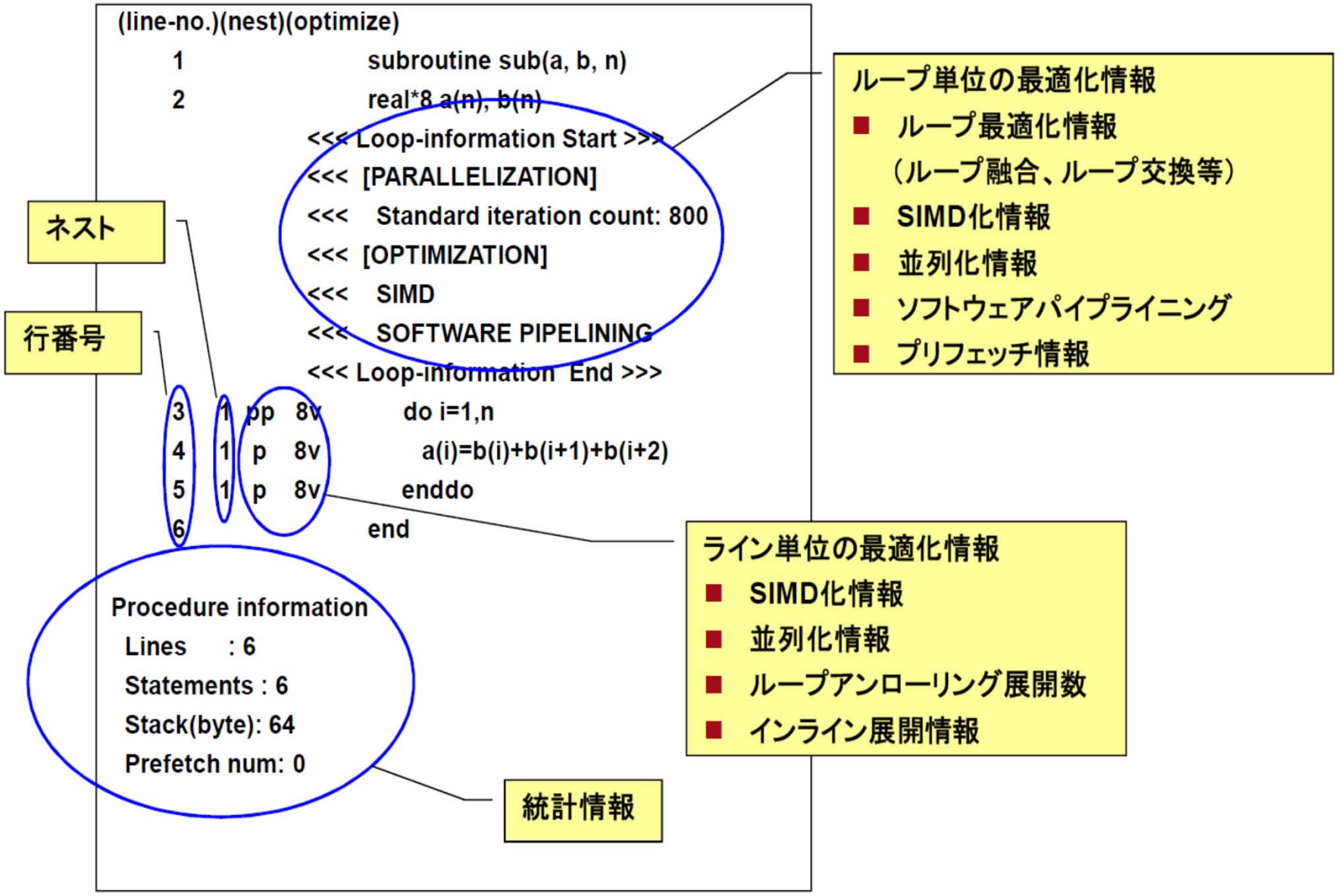
Fortran/C/C++

-Nlst=p 標準の最適化情報 (デフォルト)
-Nlst=t 詳細な最適化情報

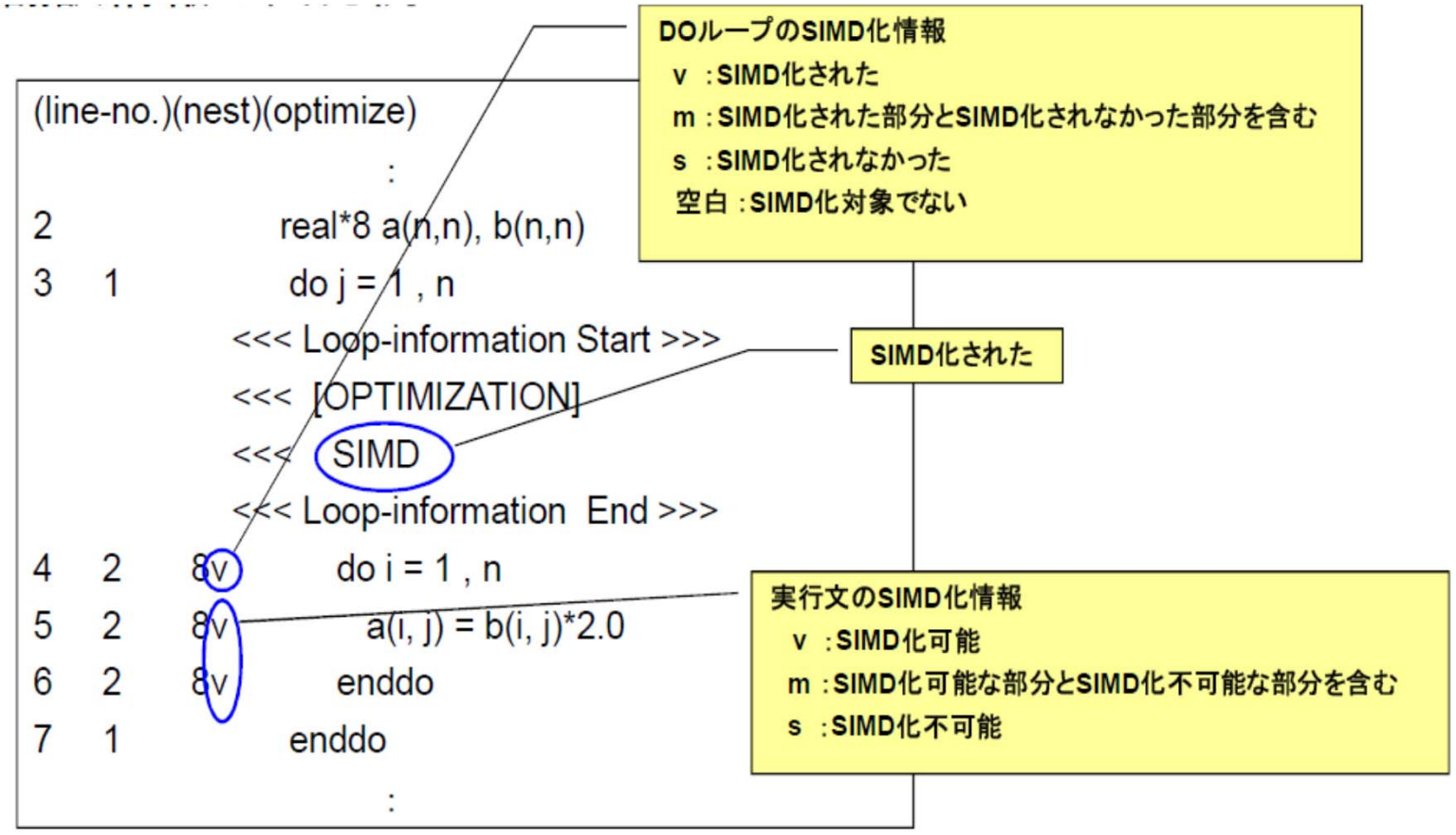
Fortran ONLY

-Nlst=a 名前の属性情報
-Nlst=d 派生型の構成情報
-Nlst=i インクルードされたファイルのプログラムリスト
　　およびインクルードファイル名一覧
-Nlst=m 自動並列化の状況をOpenMP指示文によって表現した原始プログラム出力
-Nlst=x 名前および文番号の相互参照情報

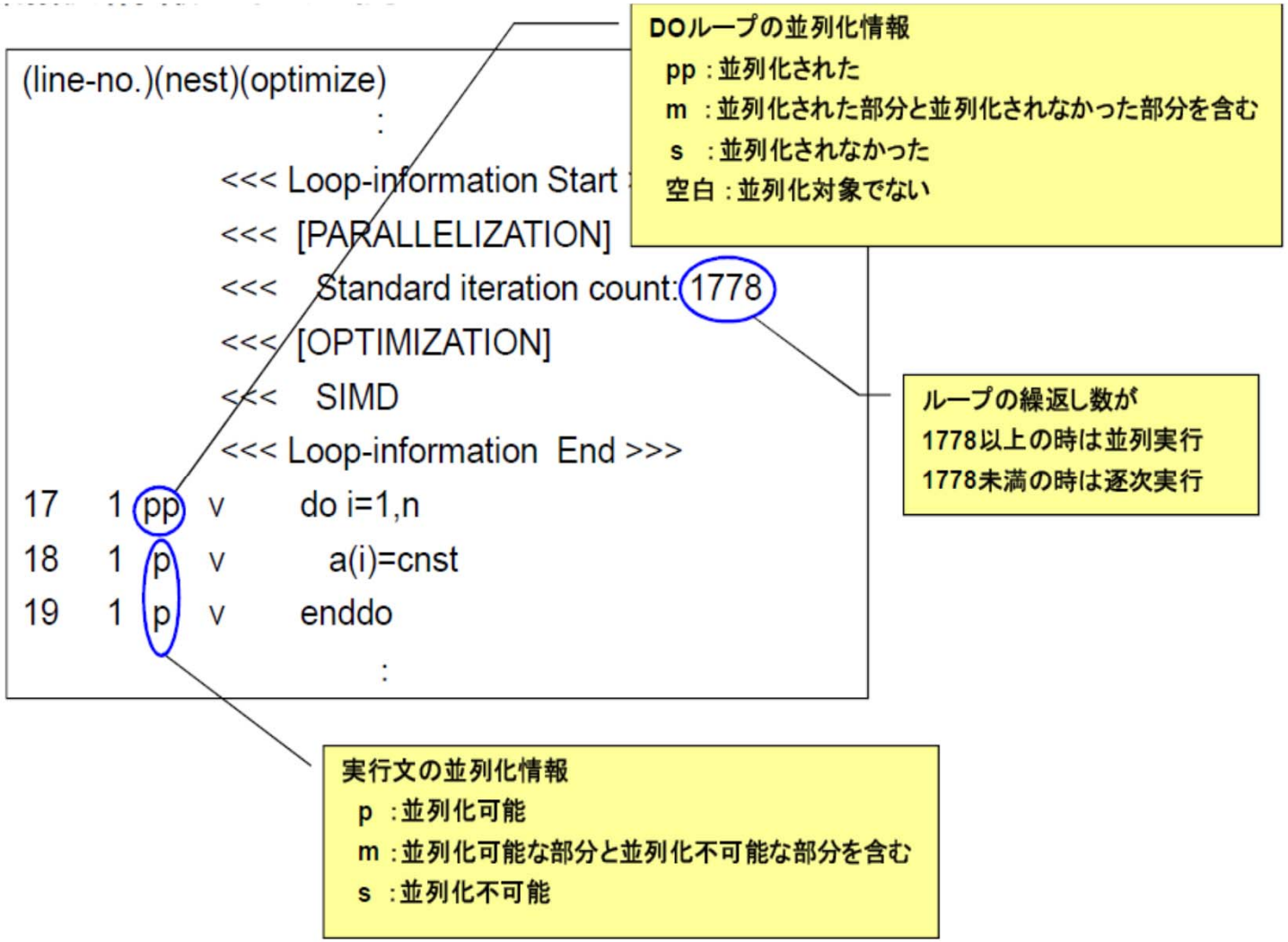
Info in *.lst


(line-no.)(nest)(optimize)
1 subroutine sub(a, b, n)
2 real*8 a(n), b(n)
<<< Loop-information Start >>>
<<< [PARALLELIZATION]
<<< Standard iteration count: 800
<<< [OPTIMIZATION]
<<< SIMD
<<< SOFTWARE PIPELINING
<<< Loop-information End >>>
3 1 pp 8v do i=1,n
4 1 p 8v a(i)=b(i)+b(i+1)+b(i+2)
5 1 p 8v enddo
6 end
Procedure information
Lines : 6
Statements : 6
Stack(byte): 64
Prefetch num: 0
ネスト
行番号
統計情報
ループ単位の最適化情報
■ ループ最適化情報
（ループ融合、ループ交換等）
■ SIMD化情報
■ 並列化情報
■ ソフトウェアパイプライニング
■ プリフェッチ情報
ライン単位の最適化情報
■ SIMD化情報
■ 並列化情報
■ ループアンローリング展開数
■ インライン展開情報

SIMD Information



Automatic Parallelization



ファイル（FX10上）

- マニュアル
  - /opt/FJSVfxlang/1.2.1/manual/japanese/Tool/Profiler_User_Guide.pdf
- Excelマクロ
  - /opt/FJSVfxlang/1.2.1/misc/CPUPA/FSDT_CPUPA.xlsm

3.5 精密PA可視化機能 (Excel) (1/4) (Precision PA Visibility Function)

Inserting Call's, Compile & Run

```
      call start_collection ( “CG” )
      S1_TIME= MPI_Wtime()
...
      E1_TIME= MPI_Wtime()
      call stop_collection ( “CG” )
```

```
      start_collection ( “CG” );
      S1_TIME= MPI_Wtime();
...
      E1_TIME= MPI_Wtime();
      stop_collection ( “CG” );
```

- Detailed profile between “start_collection” and “stop_collection” can be obtained.
- You can specify multiple segments in the program

3.5 精密PA可視化機能 (Excel) (2/4) (Precision PA Visibility Function)

Collecting Performance Data: 7X Exec's Directories: pa1~pa7, -Hpa=1~7

```
#!/bin/sh
#PJM -L "node=1"
#PJM -L "elapse=00:15:00"
#PJM -L "rscgrp=small"
#PJM -j
#PJM -o "a.lst"

export OMP_NUM_THREADS=16
fapp -C -d pa1 -Hpa=1  ./sol
fapp -C -d pa2 -Hpa=2  ./sol
fapp -C -d pa3 -Hpa=3  ./sol
fapp -C -d pa4 -Hpa=4  ./sol
fapp -C -d pa5 -Hpa=5  ./sol
fapp -C -d pa6 -Hpa=6  ./sol
fapp -C -d pa7 -Hpa=7  ./sol
```

/home/S11502/nakajima/gp.sh

```
#!/bin/sh
#PJM -L "node=1"
#PJM -L "elapse=00:15:00"
#PJM -L "rscgrp=small"
#PJM -j
#PJM -o "a.lst"
#PJM --mpi "proc=16"

export OMP_NUM_THREADS=16
fapp -C -d pa1 -Hpa=1  mpiexec ./sol
fapp -C -d pa2 -Hpa=2  mpiexec ./sol
fapp -C -d pa3 -Hpa=3  mpiexec ./sol
fapp -C -d pa4 -Hpa=4  mpiexec ./sol
fapp -C -d pa5 -Hpa=5  mpiexec ./sol
fapp -C -d pa6 -Hpa=6  mpiexec ./sol
fapp -C -d pa7 -Hpa=7  mpiexec ./sol
```

3.5 精密PA可視化機能 (Excel) (3/4) (Precision PA Visibility Function)

Performance Analysis: Transformation

```
/home/S11502/nakajima/file.sh

>$ ./file.sh
>$ ls -l ~/output_prof_*.csv
```

```
fapppx -A -d pa1 -o ~/output_prof_1.csv -tcsv -Hpa
fapppx -A -d pa2 -o ~/output_prof_2.csv -tcsv -Hpa
fapppx -A -d pa3 -o ~/output_prof_3.csv -tcsv -Hpa
fapppx -A -d pa4 -o ~/output_prof_4.csv -tcsv -Hpa
fapppx -A -d pa5 -o ~/output_prof_5.csv -tcsv -Hpa
fapppx -A -d pa6 -o ~/output_prof_6.csv -tcsv -Hpa
fapppx -A -d pa7 -o ~/output_prof_7.csv -tcsv -Hpa
```

3.5 精密PA可視化機能 (Excel) (4/4) (Precision PA Visibility Function)

Copy the files & Excel to your PC

```
>$ cd <$cur>

copy FSDT_CPUPA.xlsm to <$cur>

>$ scp sus15XX@pi.ircpi.kobe-u.ac.jp:~/output_prof_* .

start Excel
```

- MFLOPS
- Memory Throughput (GB/s)
- Instruction／命令

Programs
Spring Schoolのプログラム

Exec's	Summary
sol10	Original (L1-sol on PC)
sol20	CM Reordering
sol2x	CM Reordering + Remedy-2 for Cache Thrashing

sol10: Original (L1-sol) (C)

poi_gen

```
OLDtoNEW = (int *) allocate_vector(sizeof(int),ICELTOT);
NEWtoOLD = (int *) allocate_vector(sizeof(int),ICELTOT);

for(i=0; i<ICELTOT; i++){
    OLDtoNEW[i]= i+1;
    NEWtoOLD[i]= i+1;
}
```

solver_PCG

```
N3= N;
W = (double **)malloc(sizeof(double *)*4);
...
for(i=0; i<4; i++) {
    W[i] = (double *)malloc(sizeof(double)*N3);
...
```

sol20: +CM reordering (C)

poi_gen

```
OLDtoNEW = (int *) allocate_vector(sizeof(int),ICELTOT);
NEWtoOLD = (int *) allocate_vector(sizeof(int),ICELTOT);

RCM(ICELTOT, NL, NU, INL, IAL, INU, IAU,
    &NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW);
```

solver_PCG

```
N3= N;
W = (double **)malloc(sizeof(double *)*4);
...
for(i=0; i<4; i++) {
    W[i] = (double *)malloc(sizeof(double)*N3);
...
```

sol2x: +CM reordering + Remedy-2 (C)

poi_gen

```
OLDtoNEW = (int *) allocate_vector(sizeof(int),ICELTOT);
NEWtoOLD = (int *) allocate_vector(sizeof(int),ICELTOT);

RCM(ICELTOT, NL, NU, INL, IAL, INU, IAU,
    &NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW);
```

solver_PCG

```
N3= N+N2;                                          N2=128

W = (double **)malloc(sizeof(double *)*4);
...
for(i=0; i<4; i++) {
    W[i] = (double *)malloc(sizeof(double)*N3);
...
```

sol10: Original (L1-sol) (F)

poi_gen

```
allocate (OLDtoNEW(ICELTOT), NEWtoOLD(ICELTOT))

do i= 1, ICELTOT
  OLDtoNEW(i)= i
  NEWtoOLD(i)= i
enddo
```

solver_PCG

```
allocate (W(N,4))
```

sol20: +CM reordering (C)

poi_gen

```
allocate (OLDtoNEW(ICELTOT), NEWtoOLD(ICELTOT))

call RCM (ICELTOT, NL, NU, INL, IAL, INU, IAU,          &
          NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
```

solver_PCG

```
allocate (W(N,4))
```

sol2x: +CM reordering + Remedy-2 (F)

poi_gen

```
allocate (OLDtoNEW(ICELTOT), NEWtoOLD(ICELTOT))

call RCM (ICELTOT, NL, NU, INL, IAL, INU, IAU,          &
          NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
```

solver_PCG

```
allocate (W(N+N2,4))                                    N2=128
```

	NX=NY=NZ=128 2,097,152 meshes Load/Store= 8.28×10^{10}	NX=NY=NZ=129 2,146,689 meshes Load/Store= 8.53×10^{10}
sol10 (original)	19.50 sec. 24.11 GB/sec 13.59 %	9.15 sec. 52.98 GB/sec 3.97 %
sol20 (CM)	10.15 sec. 45.60 GB/sec 5.65 %	9.44 sec. 50.64 GB/sec 4.11 %
sol2x (CM+ Remedy-2)	9.69 sec. 47.77 GB/sec 4.20 %	9.54 sec. 50.12 GB/sec 4.11 %

Results Fortran

C ($N=128^3$)

- **sol10: 20.03 sec.**
- **sol20: 13.72**
- **sol2x: 10.05**

- **Comp. Time**
- **Memory Throughput**
- **L1D Miss Ratio (to Load/Store)**