



# OSCAR Automatic Parallelizing Compiler

## Automatic Speedup and Power Reduction



Kasahara & Kimura Lab, Waseda University, TOKYO

<http://www.kasahara.cs.waseda.ac.jp>

### OSCAR Automatic Parallelizing Compiler

To improve **effective performance**, **cost-performance** and **software productivity** and **reduce power**

#### Multigrain Parallelization

**coarse-grain parallelism** among loops and subroutines, **near fine grain parallelism** among statements in addition to **loop parallelism**

#### Data Localization

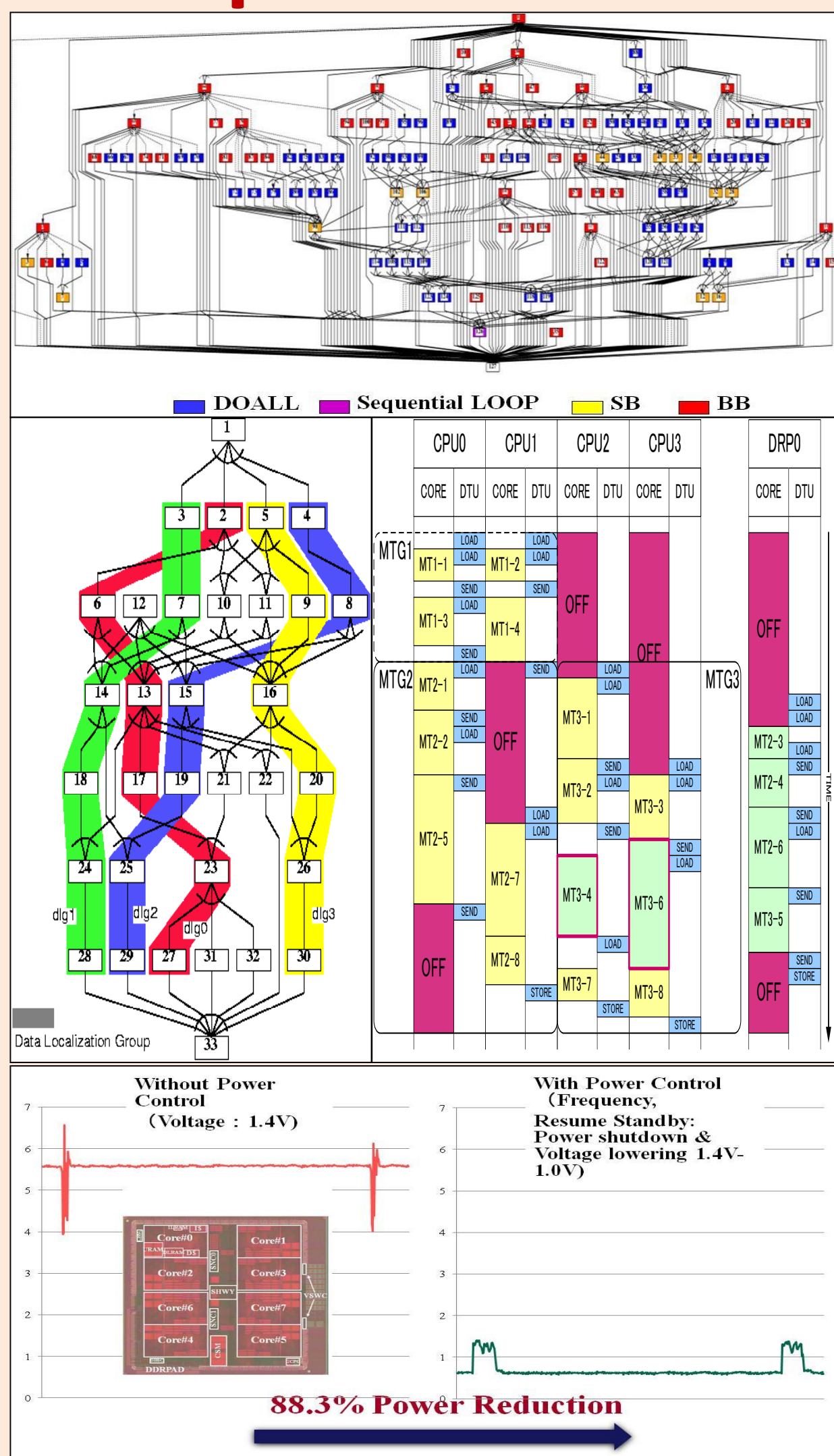
Automatic data management for **distributed shared memory**, **cache** and **local memory**

#### Data Transfer Overlapping

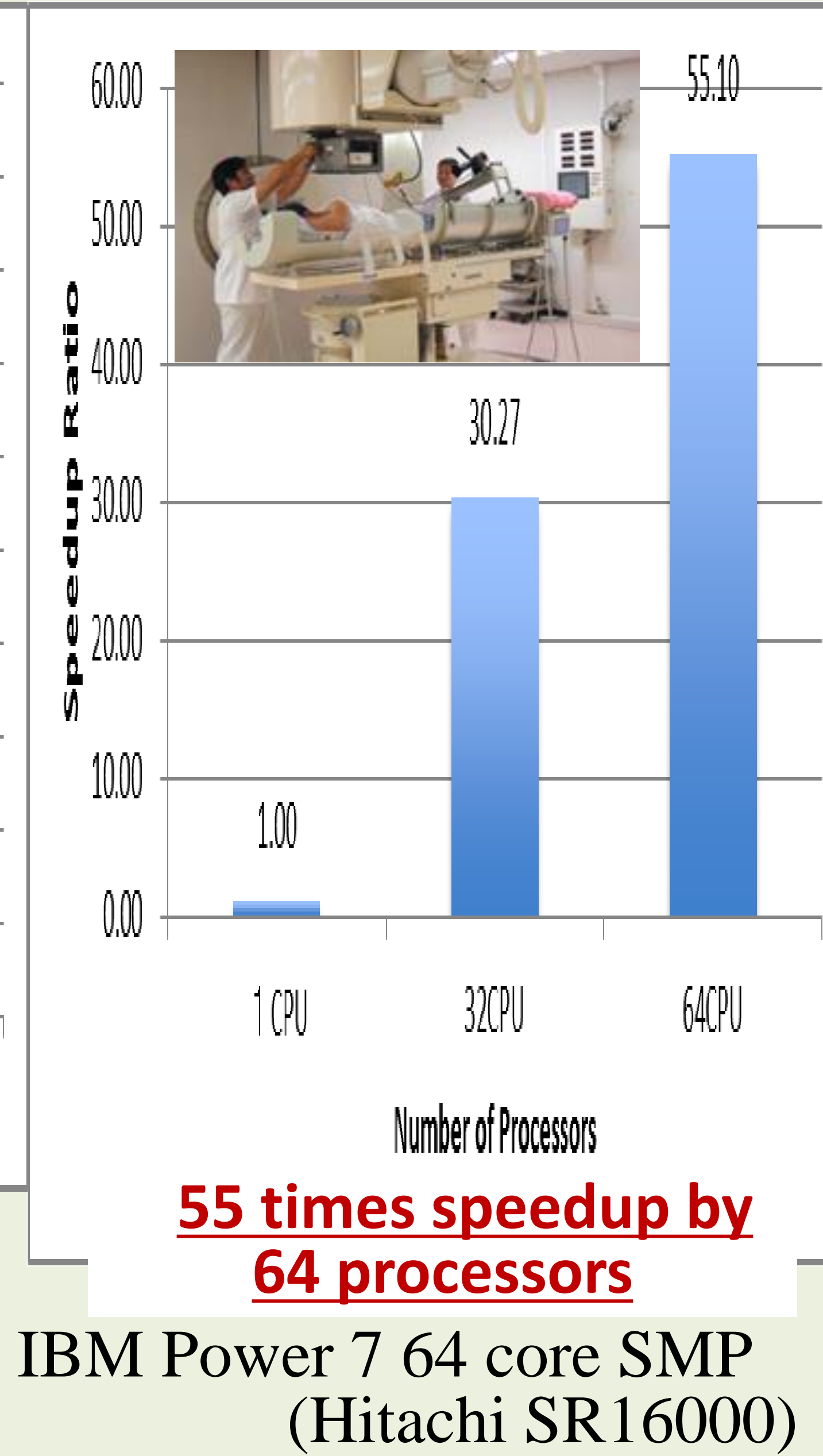
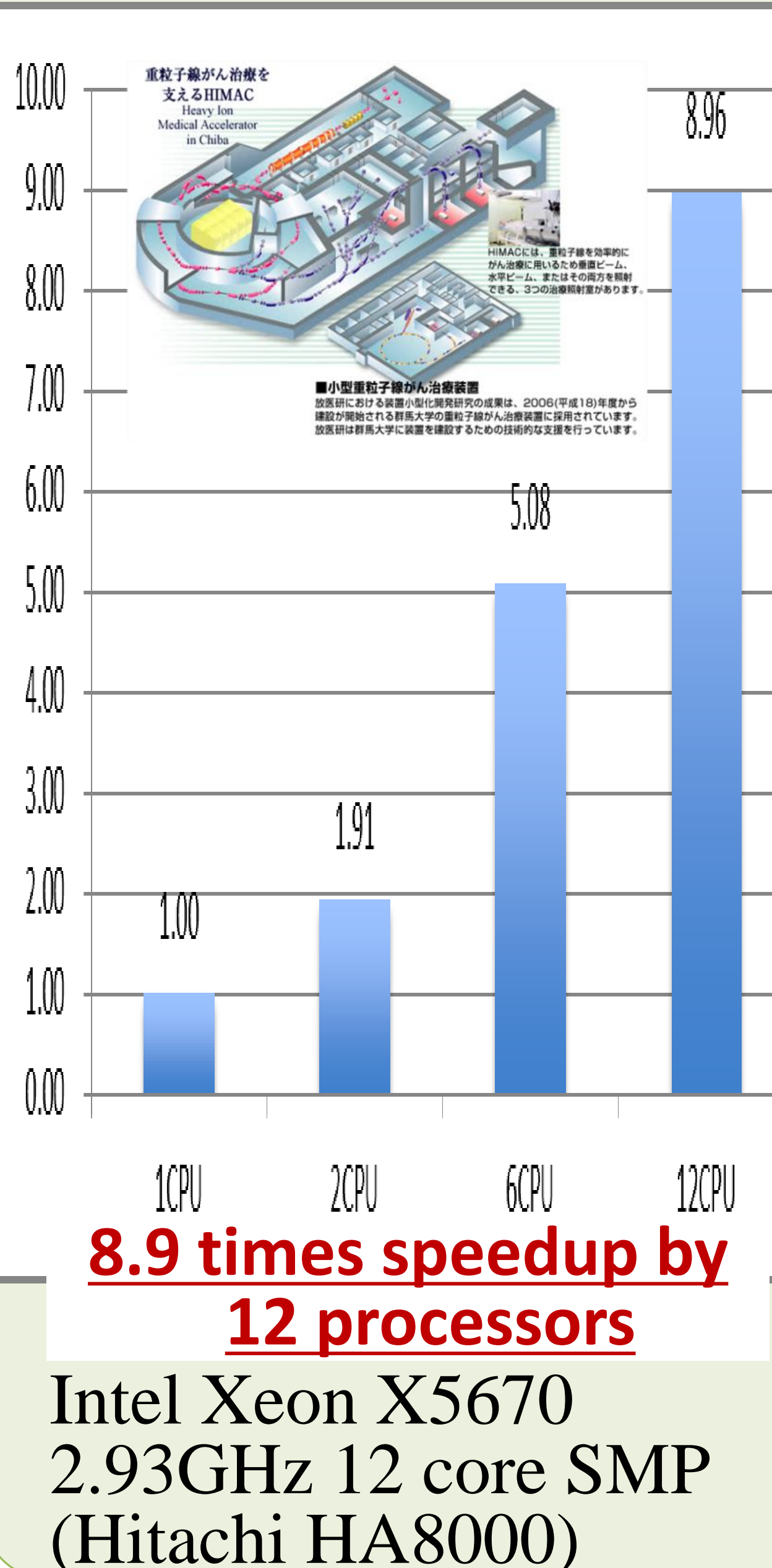
Data transfer overlapping using Data Transfer Controllers (DMAs)

#### Power Reduction

Reduction of consumed power by compiler control DVFS and Power gating with hardware supports.



### Cancer Treatment Carbon Ion Radiotherapy



### Software Coherent Cache

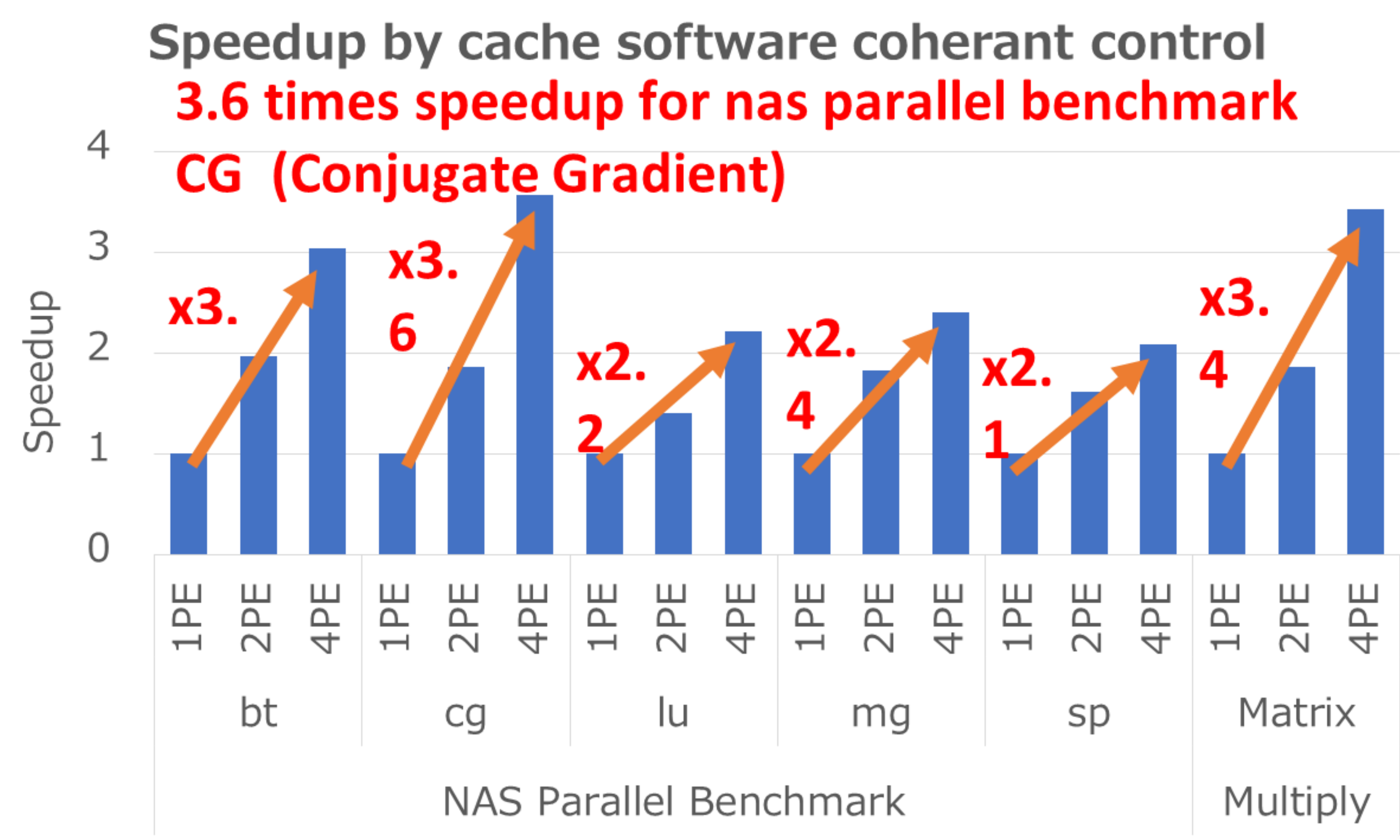
Parallelizing compiler directed software coherence technique for shared memory multicore systems without hardware cache coherence control

#### Advantages

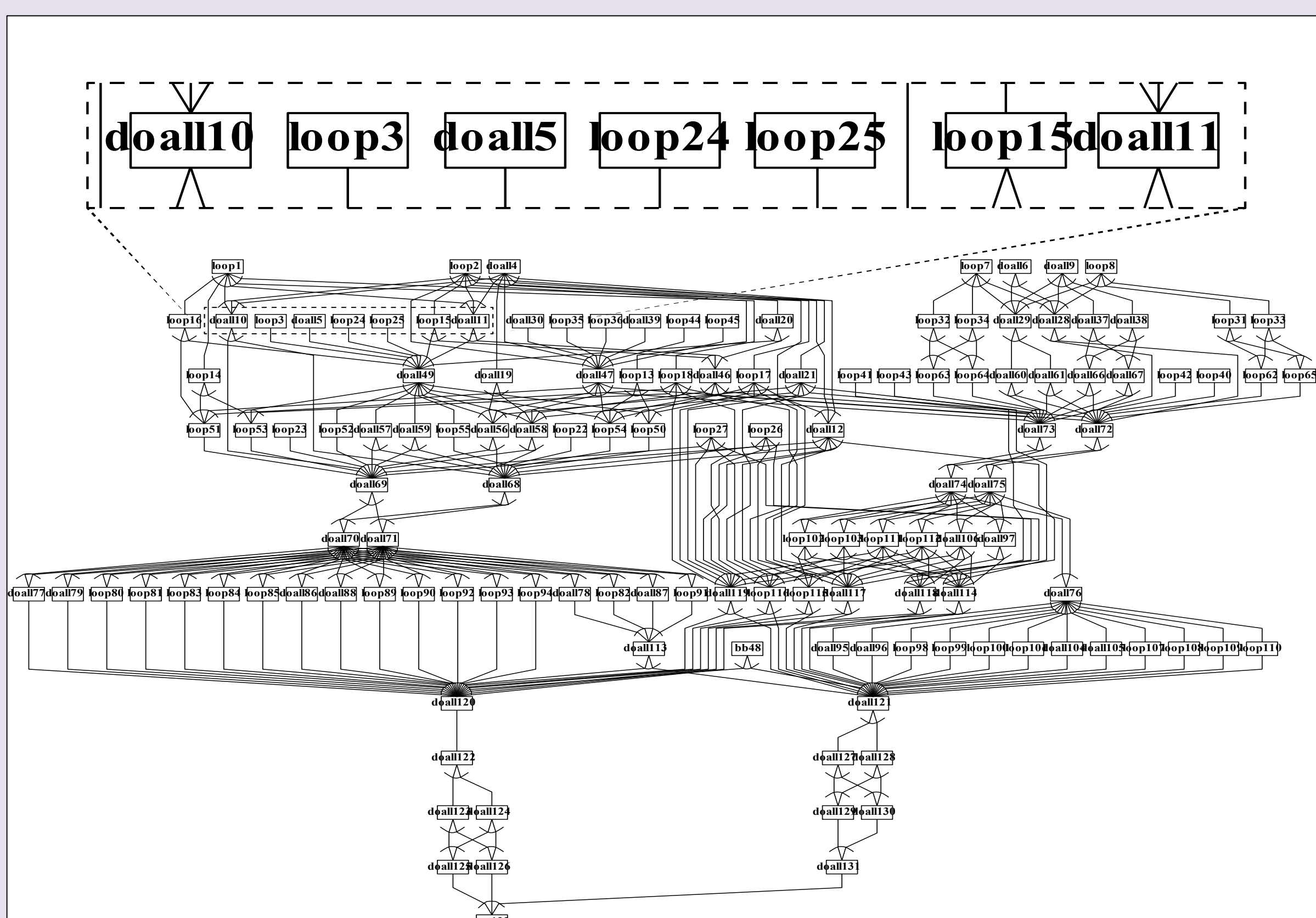
- Smaller hardware and lower power consumption brought by removing expensive hardware cache coherence mechanism
- Higher performance by compiler's careful cache operation scheduling as well as memory optimization

#### Evaluation

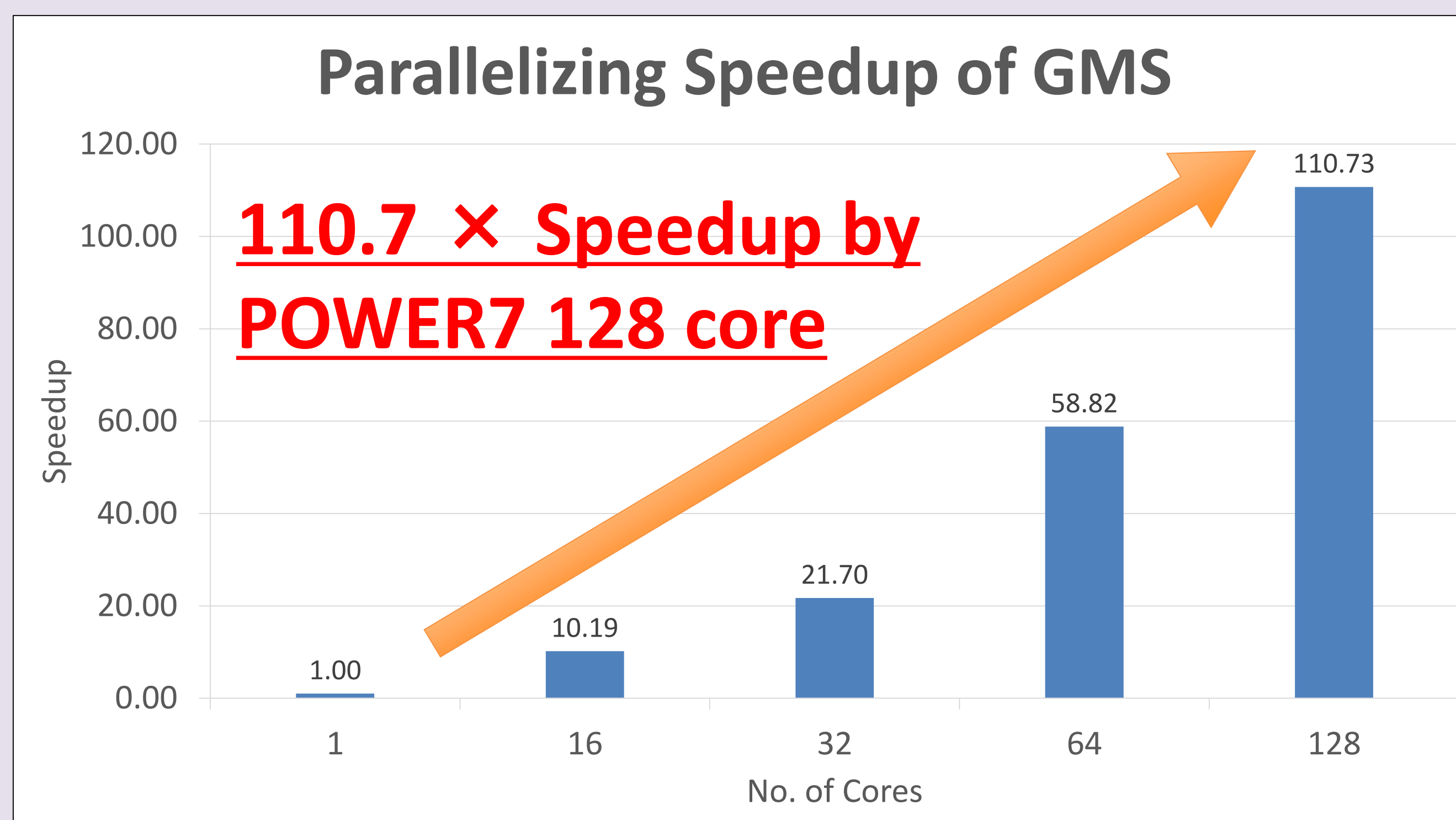
- # of PE: 1PE, 2PE, 4PE
- NIOS II multicore system implemented in Arria10 SoC FPGA
  - I\$: 32KB / D\$:32KB (Each PE)
- Application
  - NAS Parallel Benchmarks
  - Matrix Multiply (Size: 100x100)



### Parallelizing of "National Research Institute for Earth Science and Disaster Resilience" Earthquake Wave Simulation GMS by OSCAR Compiler



Task graph of OSCAR API Program



Execution environment: Hitachi SR16000 Model VM1 (IBM POWER7 Processor: 128core)

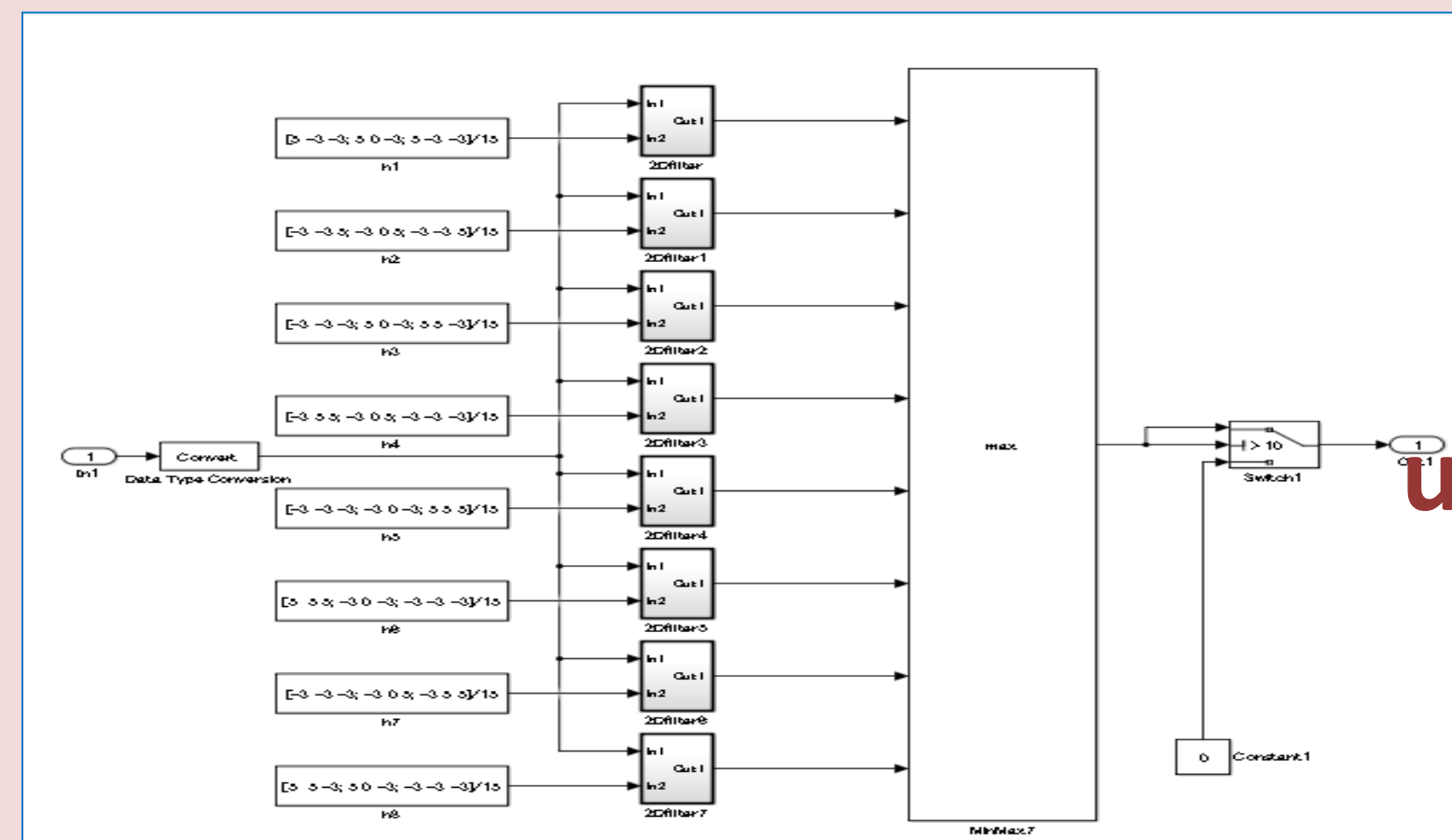


# Parallel Processing of MATLAB/Simulink by OSCAR Compiler on Intel, ARM & Renesas multi cores

Kasahara & Kimura Lab, Waseda University, TOKYO

- OSCAR Compiler
  - MATLAB/Simulink
  - Multi grain Parallelization
- <http://www.kasahara.cs.waseda.ac.jp>

## Automatic Parallelization of MATLAB/Simulink by OSCAR Compiler



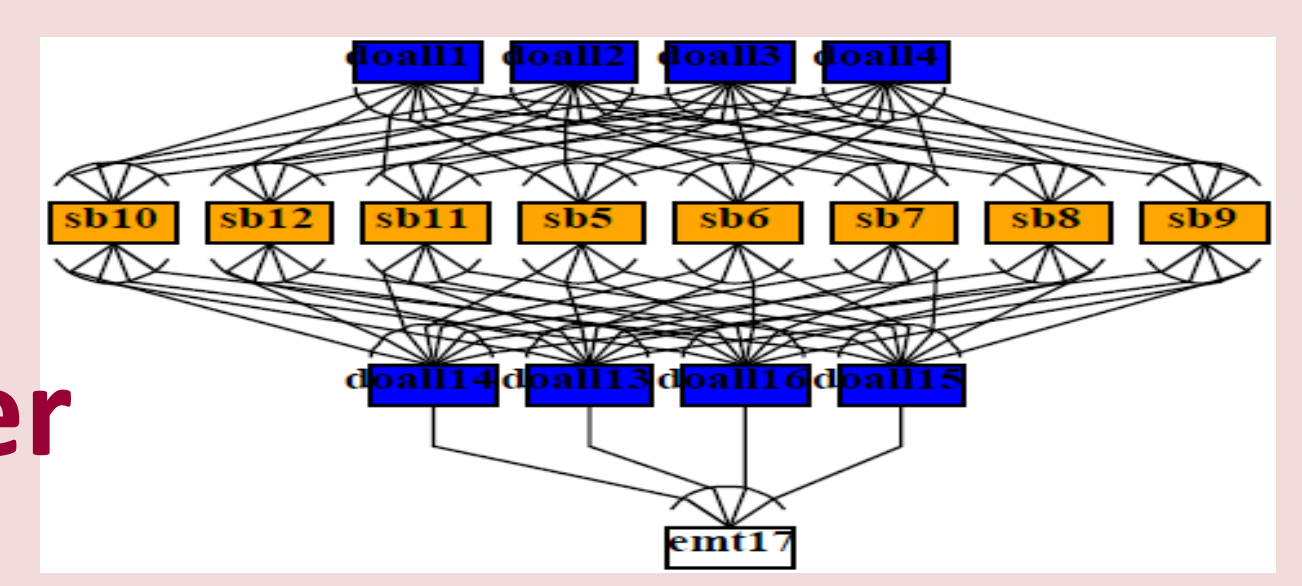
Generate C code using Embedded Code

```

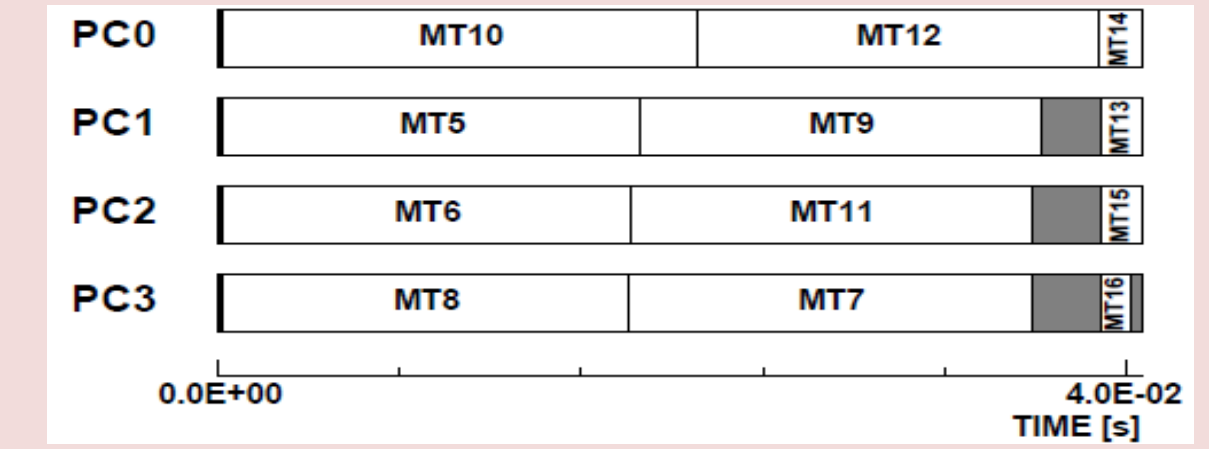
/* Model step function */
void VesselExtraction_step(void)
{
    int32_T i;
    real_T u0;

    /* DataTypeConversion: '<S1>/Data Type Conversion' incorporates:
    * Import: '<Root>/In1'
    */
    for (i = 0; i < 16984; i++) {
        VesselExtraction_B.DataTypeConversion[i] = VesselExtraction_U.In1[i];
    }
    /* End of DataTypeConversion: '<S1>/Data Type Conversion' */
    /* Outputs for Atomic SubSystem: '<S1>/2Dfilter' */
    constant: '<S1>/h1' */
    VesselExtraction_DFilter(VesselExtraction_B.DataTypeConversion,
    VesselExtraction_P.h1_Value, &VesselExtraction_B.DFilter,
    (P_Filter_VesselExtraction_T *)&VesselExtraction_P.DFilter);
    /* End of Outputs for SubSystem: '<S1>/2Dfilter' */
    /* Outputs for Atomic SubSystem: '<S1>/2Dfilter1' */
    constant: '<S1>/h2' */
    VesselExtraction_DFilter(VesselExtraction_B.DataTypeConversion,
    VesselExtraction_P.h2_Value, &VesselExtraction_B.DFilter1,
    (P_Filter_VesselExtraction_T *)&VesselExtraction_P.DFilter1);
}
    
```

OSCAR Compiler



(1) Generate MTG → Parallelism



(2) Generate Gantt chart → Scheduling in a multicore

```

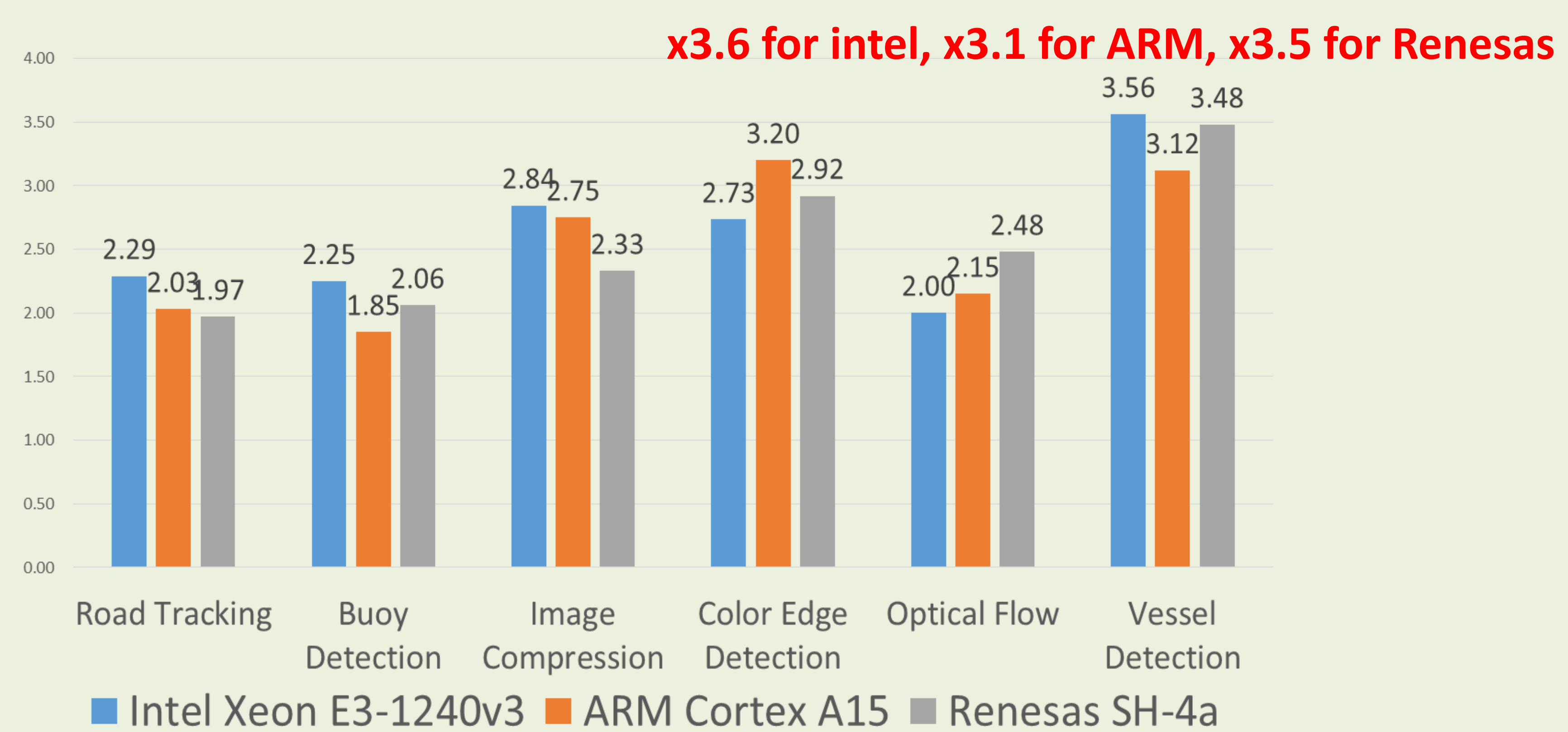
void VesselExtraction_step ( )
{
    int thr1 ;
    int thr2 ;
    int thr3 ;
    {
        oscar_thread_create ( & thr1 ,
        thread_function_001 , (void*)1 ) ;
        oscar_thread_create ( & thr2 ,
        thread_function_002 , (void*)2 ) ;
        oscar_thread_create ( & thr3 ,
        thread_function_003 , (void*)3 ) ;

        VesselExtraction_step_PEO ( ) ;

        oscar_thread_join ( thr1 ) ;
        oscar_thread_join ( thr2 ) ;
        oscar_thread_join ( thr3 ) ;
    }
}
    
```

(3) Generate parallelized C code using the OSCAR API → Multiplatform execution (Intel, ARM and SH etc)

### Speedups of MATLAB/Simulink Image Processing on Various 4core Multicores (Intel Xeon, ARM Cortex A15 and Renesas SH4A)



Road Tracking, Image Compression : <http://www.mathworks.co.jp/help/vision/examples>  
 Buoy Detection : <http://www.mathworks.co.jp/matlabcentral/fileexchange/44706-buoy-detection-using-simulink>  
 Color Edge Detection : <http://www.mathworks.co.jp/matlabcentral/fileexchange/28114-fast-edges-of-a-color-image--actual-color--not-converting-to-grayscale/>  
 Vessel Detection : <http://www.mathworks.co.jp/matlabcentral/fileexchange/24990-retinal-blood-vessel-extraction/>



## Vector Processing of Parallelized Program by OSCAR Compiler on NEC SX-Aurora TSUBASA

- OSCAR Compiler
- SX-Aurora TSUBASA

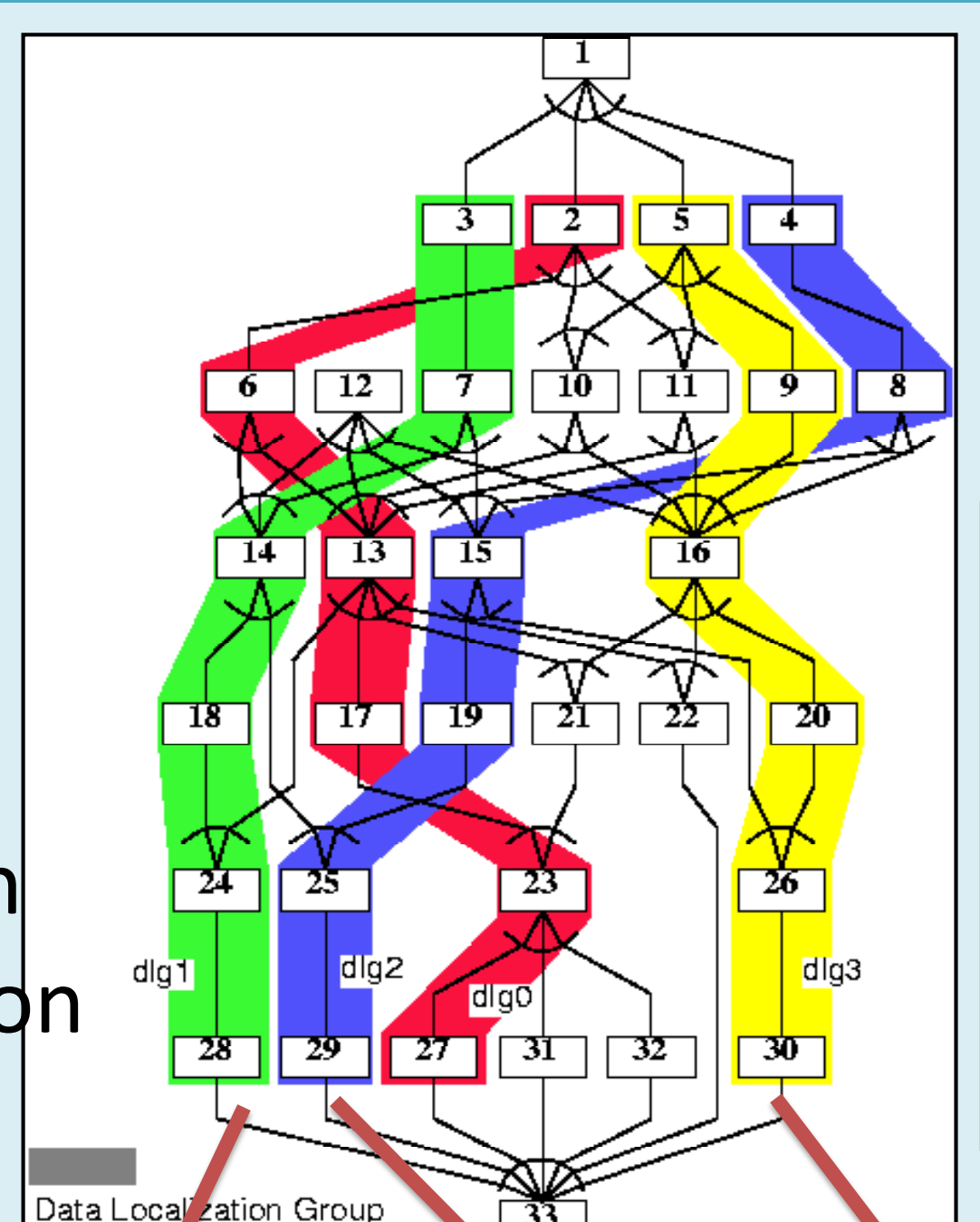
```

DO 100 J=1,N
DO 100 I=1,M
C(I,J+1) = 5D0*(C(I+1,J)+C(I,J))*C(I+1,J)
C(I,J+1) = 5D0*(C(I+1,J)+C(I,J))*C(I+1,J)
C(I+1,J+1) = (FSDY*(C(I+1,J+1)+C(I,J)))*C(I+1,J+1)
C(I+1,J+1) = (FSDY*(C(I+1,J+1)+C(I,J)))*C(I+1,J+1)
C(I,J) = (C(I,J)+25D0*(C(I+1,J)+C(I-1,J))*C(I,J))
C(I,J) = (C(I,J)+25D0*(C(I+1,J)+C(I-1,J))*C(I,J))
100 CONTINUE

DO 200 J=1,N
DO 200 I=1,M
PNEW(I+1,J) = UOLD(I+1,J)+
1 TDTS*(C(I+1,J+1)+C(I,J))*C(I+1,J)+C(I+1,J)+C(I,J)
PNEW(I+1,J) = UOLD(I+1,J)+
2 TDTS*(C(I+1,J+1)+C(I,J))*C(I+1,J)+C(I+1,J)+C(I,J)
PNEW(I,J+1) = UOLD(I,J+1)+
1 TDTS*(C(I+1,J+1)+C(I,J))*C(I+1,J+1)+C(I+1,J+1)+C(I,J)
PNEW(I,J+1) = UOLD(I,J+1)+
2 TDTS*(C(I+1,J+1)+C(I,J))*C(I+1,J+1)+C(I+1,J+1)+C(I,J)
1 TDTS*(C(I+1,J+1)+C(I,J))*C(I+1,J+1)+C(I+1,J+1)+C(I,J)
1 TDTS*(C(I+1,J+1)+C(I,J))*C(I+1,J+1)+C(I+1,J+1)+C(I,J)
200 CONTINUE

DO 300 J=1,N
DO 300 I=1,M
UOLD(I,J) = U(I,J)+ALPHA*(PNEW(I,J)-2.*U(I,J)+UOLD(I,J))
VOLD(I,J) = V(I,J)+ALPHA*(PNEW(I,J)-2.*V(I,J)+VOLD(I,J))
PNEW(I,J) = P(I,J)+ALPHA*(PNEW(I,J)-2.*P(I,J)+POLD(I,J))
U(I,J) = PNEW(I,J)
V(I,J) = PNEW(I,J)
P(I,J) = PNEW(I,J)
300 CONTINUE
swim
    
```

OSCAR Compiler  
Automatic Coarse grain Parallelization



```

PROGRAM OSCAR_MAIN
$OMP PARALLEL SECTIONS
$OMP SECTION
CALL SHALOW_PEO
$OMP SECTION
CALL SHALOW_PE1
$OMP SECTION
CALL SHALOW_PE2
$OMP SECTION
CALL SHALOW_PE3
$OMP END PARALLEL SECTIONS
END
    
```

NEC Compiler

Automatic Vectorization

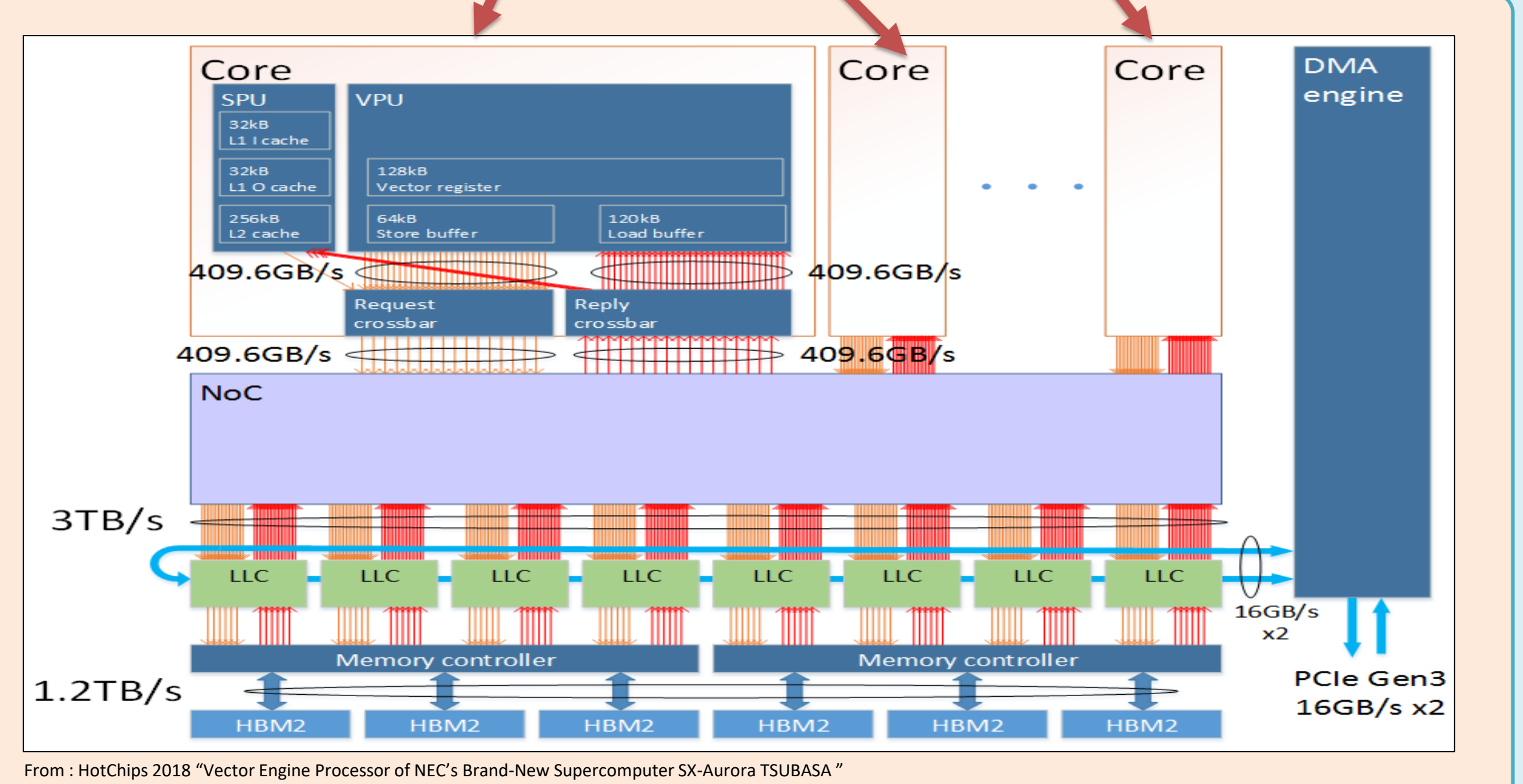
Executable File

SPEC benchmark 171.swim

NUMBER OF POINTS IN THE X DIRECTION	512
NUMBER OF POINTS IN THE Y DIRECTION	512
GRID SPACING IN THE X DIRECTION	25000.
GRID SPACING IN THE Y DIRECTION	25000.
TIME STEP	20.
TIME FILTER PARAMETER	0.001
NUMBER OF ITERATIONS	10
Pcheck =	0.1311E+11
Ucheck =	0.5215E+05
Vcheck =	0.5215E+05

Parallelize & Vectorize execution

**SX-Aurora TSUBASA Architecture**  
 8 Vector Core  
 16MB LLC  
 DMA Engine  
 6 HBM2 Controller



### Speedups on SX-Aurora TSUBASA swim from SPEC2000 relative performance

