Implementation and Evaluation of OpenMP for Hitachi SR8000

Yasunori Nishitani, Kiyoshi Negishi, Eiji Nunohiro
Software Division, Hitachi, Ltd.

Hiroshi Ohta
Systems Development Laboratory, Hitachi, Ltd.

HITACHI

Outline

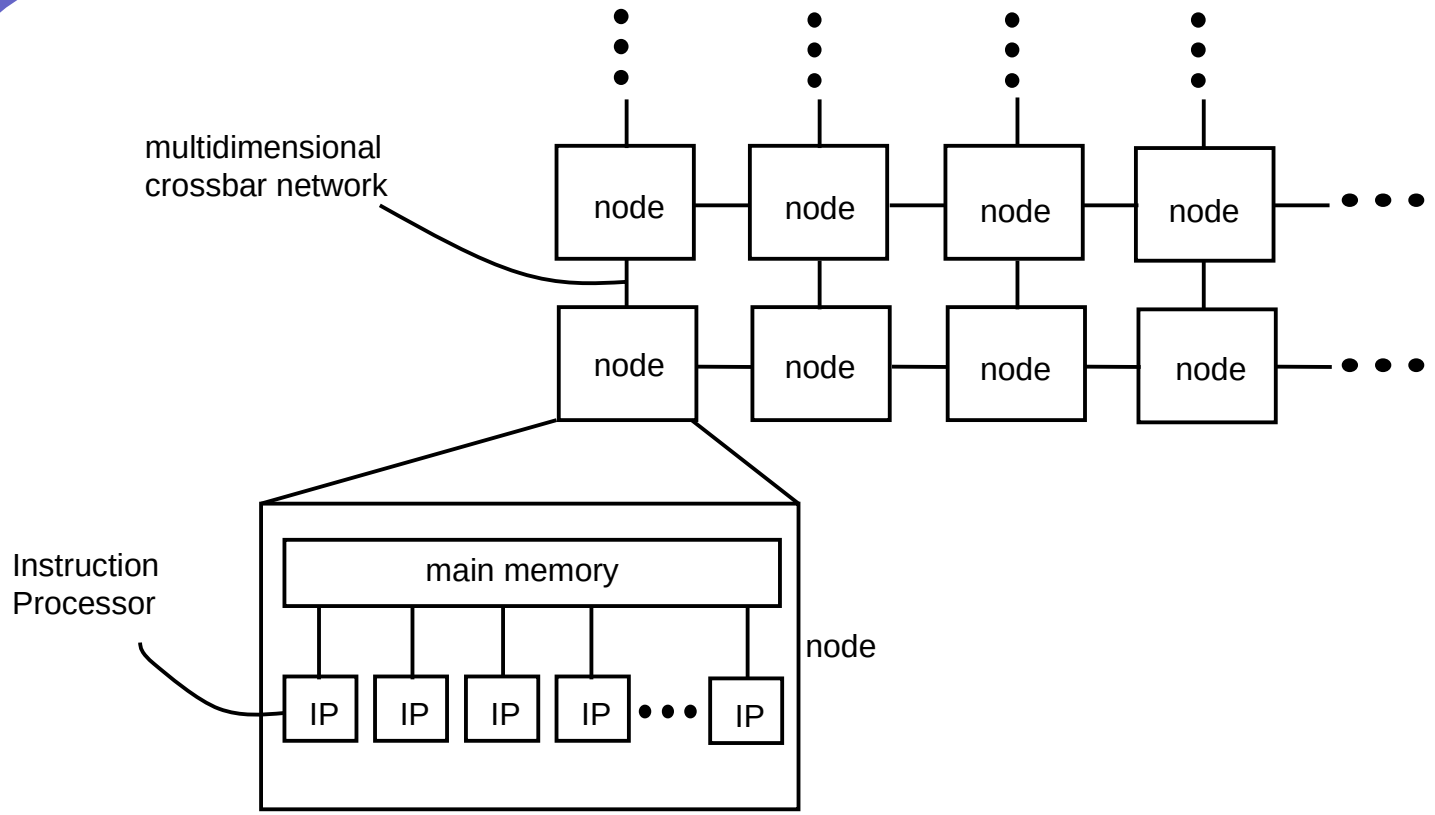
- Background
- SR8000 System
- Compiler Overview
- Implementation
- Benchmark Result
- Conclusions

Background

- Parallelization within the node of SR8000
  - shared memory parallelization
  - parallelized by
    - automatic parallelization of the compiler
    - *poption directive(Hitachi proprietary)

  ⇨ portability is not good
- OpenMP
  - specifies shared memory parallelism
  - increases portability
    - industrial standard
    - parallelism is explicitly specified
  - orphaned directives
  - dynamic scheduling

SR8000 Architecture



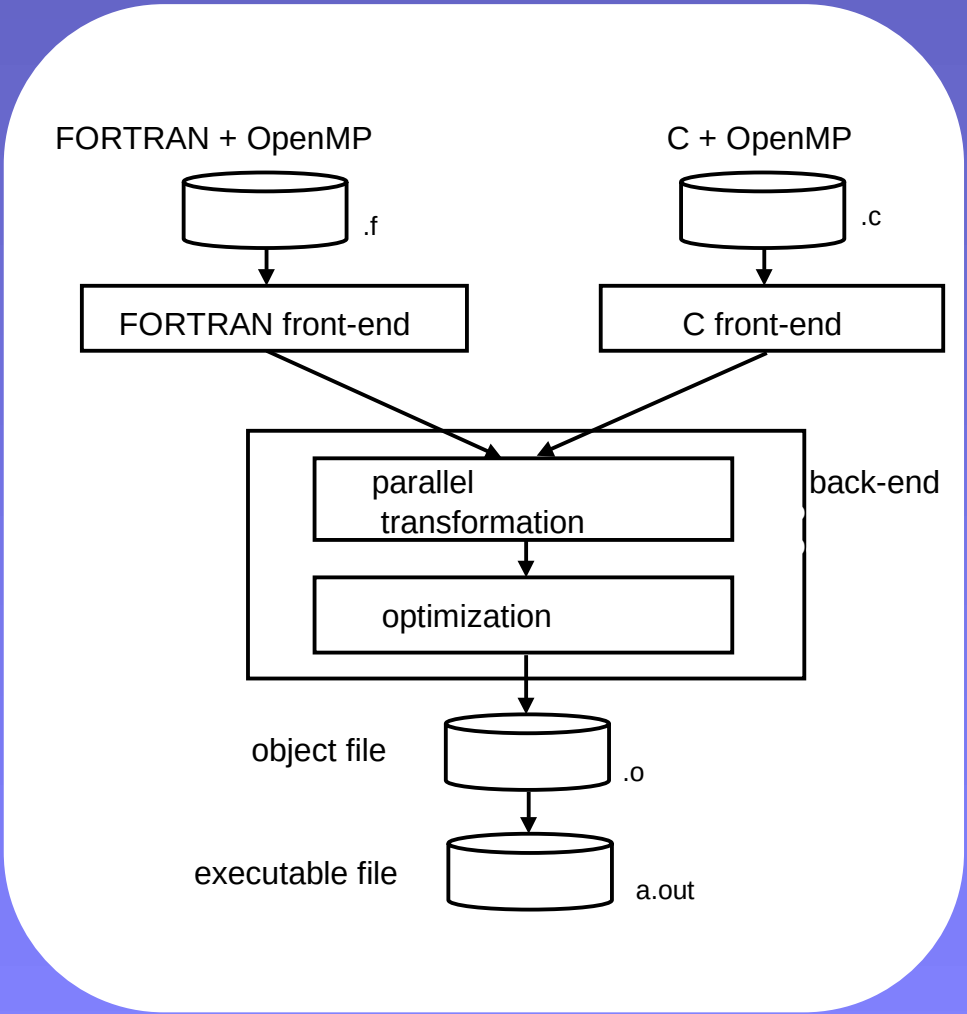
- peak performance - 12 GFlops/node(model F1)
- scalable up to 6.1 TFlops
- interconnect bandwidth - 1 GB/s (single direction) x 2

Programming Model

- Inter-node
  - distributed memory parallelism
  - MPI, Remote-DMA, HPF
- Intra-node
  - shared memory parallelism
  - automatic parallelization, *poption, OpenMP

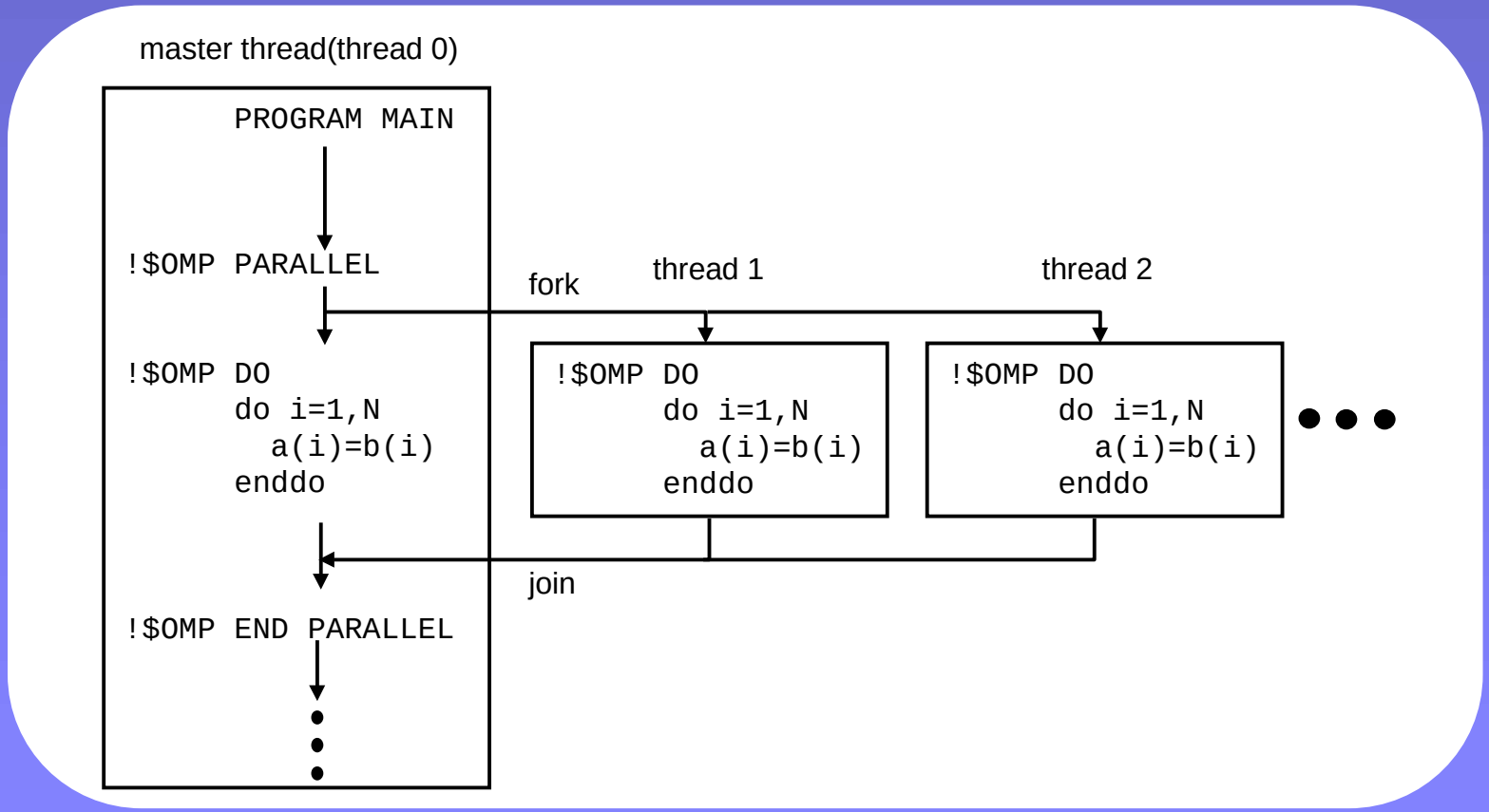
Compiler Overview

- Fortran and C is supported
- Full set of OpenMP API 1.0 is supported
- Not implemented
  - true nested parallelism
  - dynamic thread adjustment
- Can be mixed with procedures compiled by automatic parallelization of the compiler



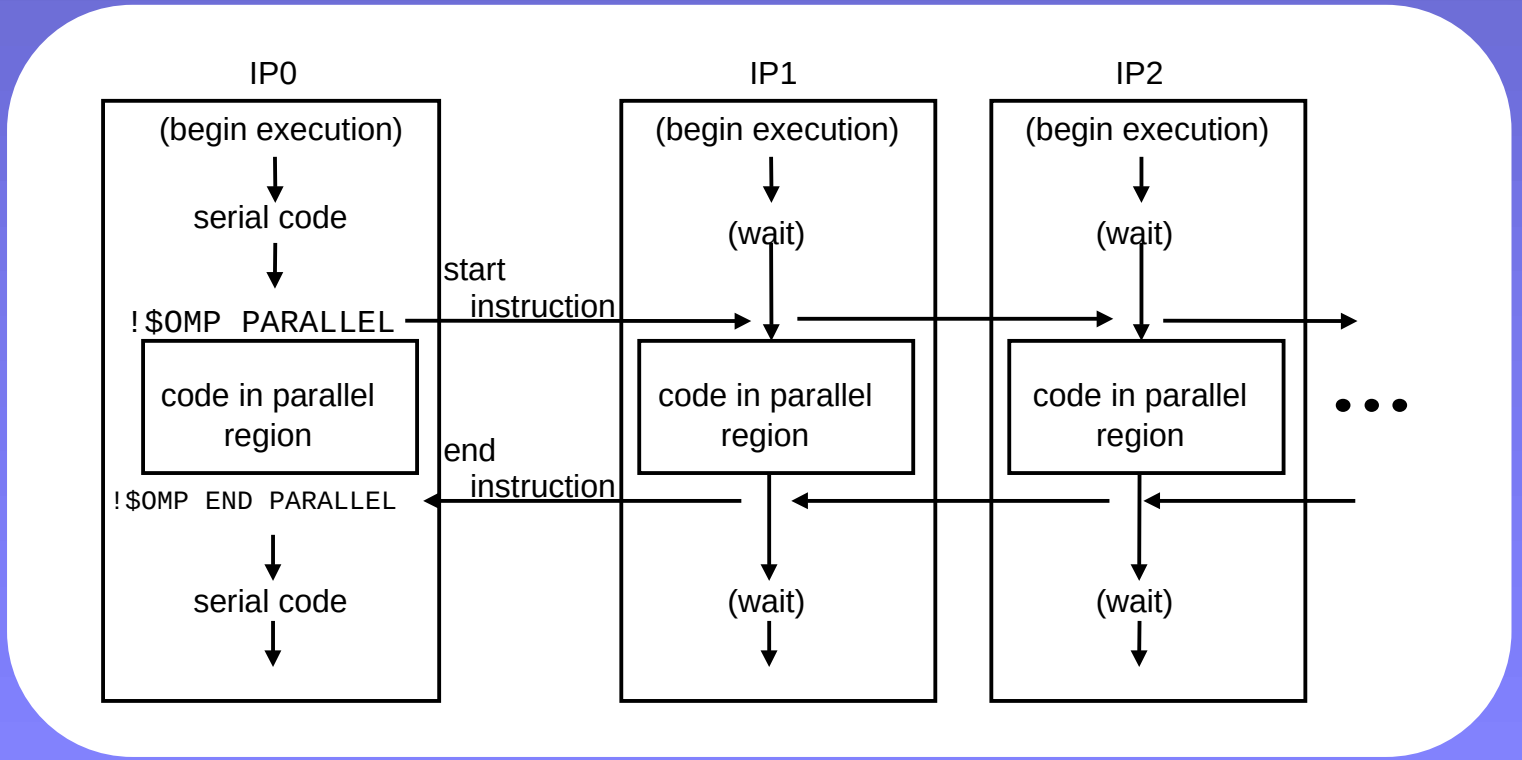
PARALLEL Region

- Execution model of OpenMP
  - fork-join model
  - PARALLEL directive controls thread creation
  - work-sharing directive divides tasks into threads



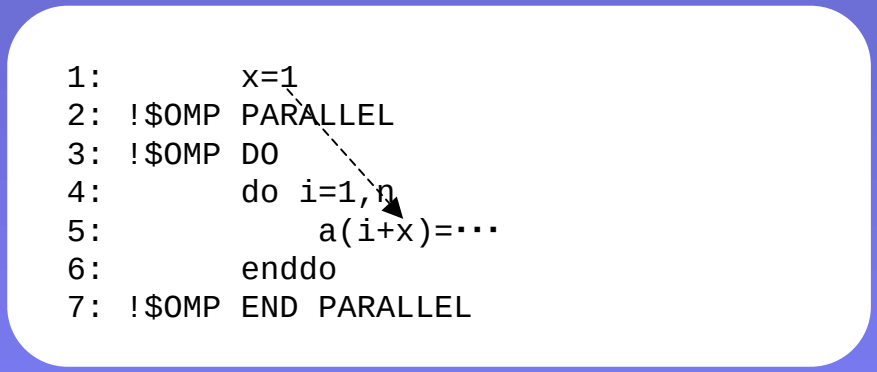
Implementation of PARALLEL

- Implemented using COMPAS(Co-Operative Microprocessors with single Adress Space)
- Threads start fast by hardware instruction
- Threads can bind to fixed IP (no scheduling overhead)



PARALLEL and Optimization

- PARALLEL region is extracted to an internal procedure inside the compiler
- Optimization should be done across PARALLEL region boundary



- Not all optimizations for serial programs are legal for parallel programs

PARALLEL and Optimization

- Certain optimizations are performed before extracting PARALLEL region
- Dummy reference is inserted at the PARALLEL directive to avoid illegal optimization

```
1:        x=1
2: !$OMP PARALLEL REDUCTION(x)  <-- dummy def & use of x
3:        x=x+f(a)
4: !$OMP END PARALLEL           <-- dummy def & use of x
5:        y=x
6:        ...
```

reference of x should not be moved to outside of the PARALLEL region

DO directive

- Loop parallelization
  - static and dynamic scheduling are implemented
- Optimization
  - moves loop index range calculation codes to outer loop

```
1:        do j=1,M
2:  !$OMP DO
3:          do i=1,N
4:            ...
5:          enddo
6:        enddo
```

```
1:        do j=1,M
2:          lb=1+blksz*threadno
3:          ub=min(lb+blksz-1,N)
4:          do i=lb,ub
5:            ...
6:          enddo
7:        enddo
```

  - performance improvement by -procnum=8 option
    - loop index range calculation is simplified
    - exploit maximum performance of the 8IPs in the node

Diagnostic Messages

- Provides diagnostic messages for parallelization
- Useful for
  - discovering incorrect directives
  - converting a serial program to OpenMP program

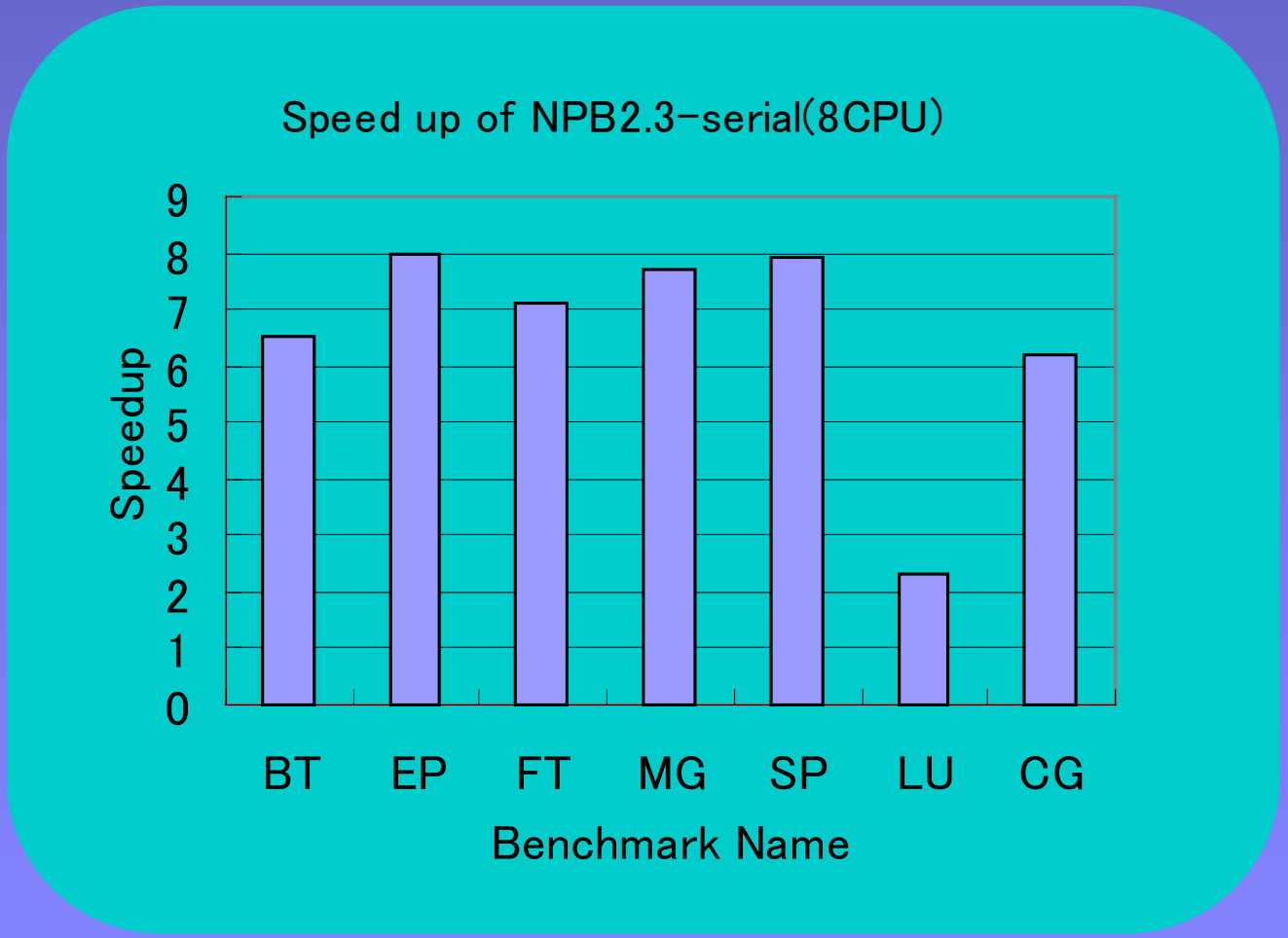
```
1:       subroutine sub(a,c,n,m)
2:       real a(n,m)
3: !$OMP PARALLEL DO PRIVATE(tmp1)
4:       do j=2,m
5:         do i=1,n
6:           jm1=j-1
7:           tmp1=a(i,jm1)
8:           a(i,j)=tmp1/c
9:         enddo
10:      enddo
11:      end

(diagnosis for loop structure)
   KCHF2015K
             the do loop is parallelized by "omp do" directive. line=4
   KCHF2200K
             the parallelized do loop contains data dependencies across loop
             iterations. name=A line=4
   KCHF2201K
             the variable or array in do loop is not privatized. name=JM1 line=4
```

Evaluation

- NPB2.3-serial benchmark (class A)
- Parallelized by only inserting OMP directives
  - except loops which cannot be parallelized by OMP directive
  - no serial and parallel tunings
- Speedup for 8 processors are measured

Performance

- Speedup of NPB2.3-serial Benchmark(class A)

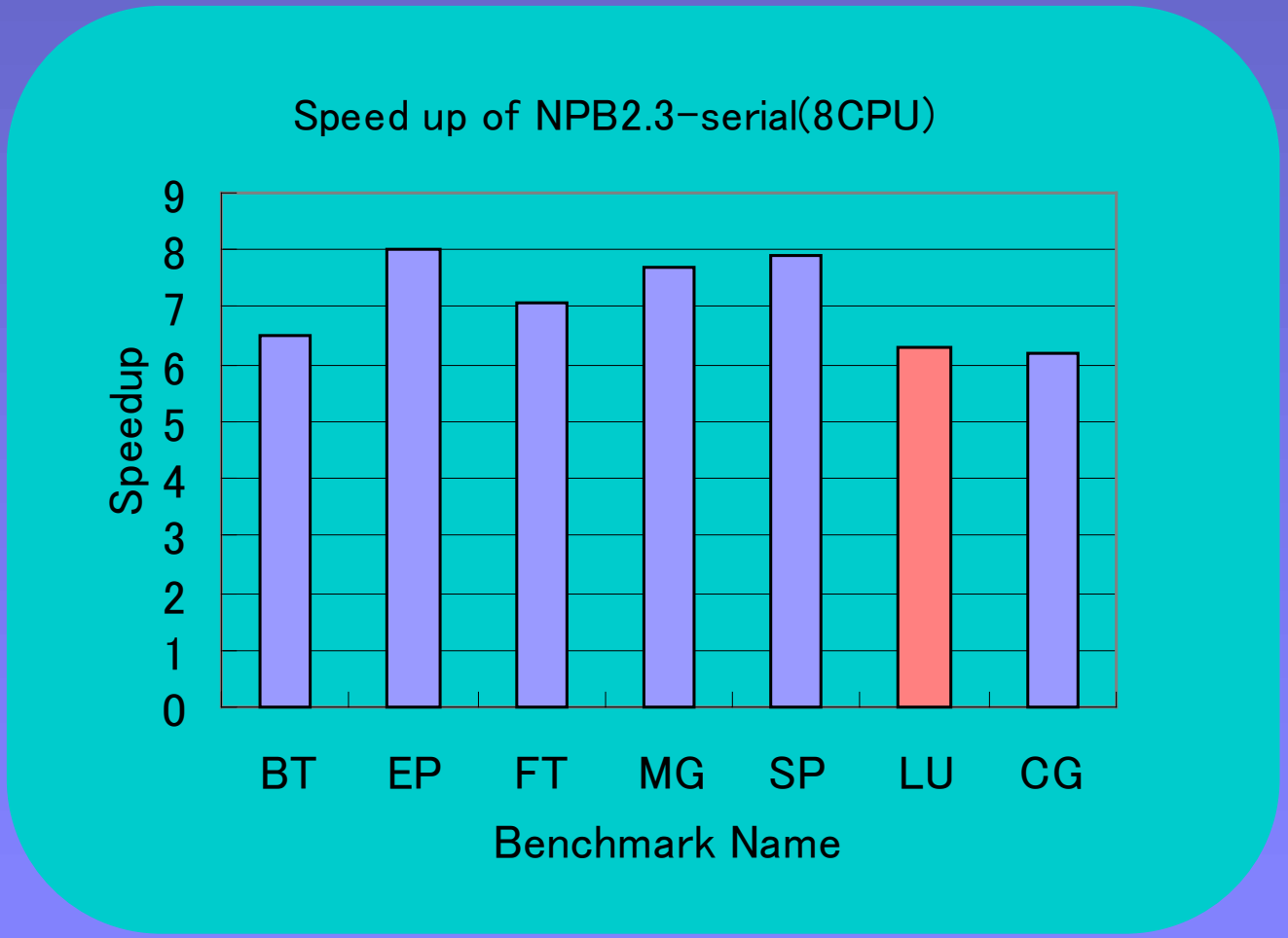


Performance

- speedup on 8 processors is between 6.2 to 8.0
  except LU
- LU has wavefront style loop which can't be parallelized by OpenMP directive
  - speedup of 6.3 is achieved by combining automatic parallelization and OpenMP

Performance

- Speedup of NPB2.3-serial Benchmark(class A)



Problems of OpenMP

- some loops cannot be parallelized by OpenMP directives
  - wave-front style loop

```
1:      do j=jst,jend
2:        do i=ist,iend
3:          do m=1,5
4:            v(m,i,j,k) = v(m,i,j,k)
5:            - omega*(ldy(m,1,i,j)*v(1,i,j-1,k)
6:     >               +ldx(m,1,i,j)*v(1,i-1,j,k)
7:     >               + ...
```

  - array reduction
  - loop containing induction variable

```
1:      K=...
2:      do I=1,N
3:        A(2*I-1)=K
4:        A(2*I)=K+3
5:        K=K+6
6:      enddo
```

Conclusions

- OpenMP for SR8000
  - Fast thread execution using COMPAS
  - Optimization across PARALLEL region boundaries
  - Optimization for loop parallelization
  - Support for parallelization diagnostic messages
- Scalability
  - 6.2 to 8.0 speedup by NPB2.3 benchmark