Research and Development in Advanced Parallelizing Compiler Technology Project

APC Project Leader

Prof. Hironori Kasahara

Waseda University

http://www.kasahara.elec.waseda.ac.jp

kasahara@waseda.jp

http://www.apc.waseda.ac.jp

Needs for Advanced Parallelizing Compiler

- **Wide use of multiprocessor architectures**
  - From chip multiprocessor to high performance computer such as
    - » IBM Power4 chip multiprocessor
    - » Sun Ultra80 4processor multiprocessor workstation
    - » Sun V880 8 processor entry level server
    - » IBM pSeries690 high end sever (RegattaH)
    - » Earth Simulator world fastest supercomputer

http://www.sun.com/desktop/products/ultra80/

  - Increasing gap between peak and effective performance with increase of the number of processors
  - Increasing speed gap between processor and memory
  - Difficulty in parallel programming and tuning
- **Compilers to improve Effective Performance, Cost Performance and Ease of Use (Software Productivity) are required.**

IBM Power4 Chip Multiprocessor

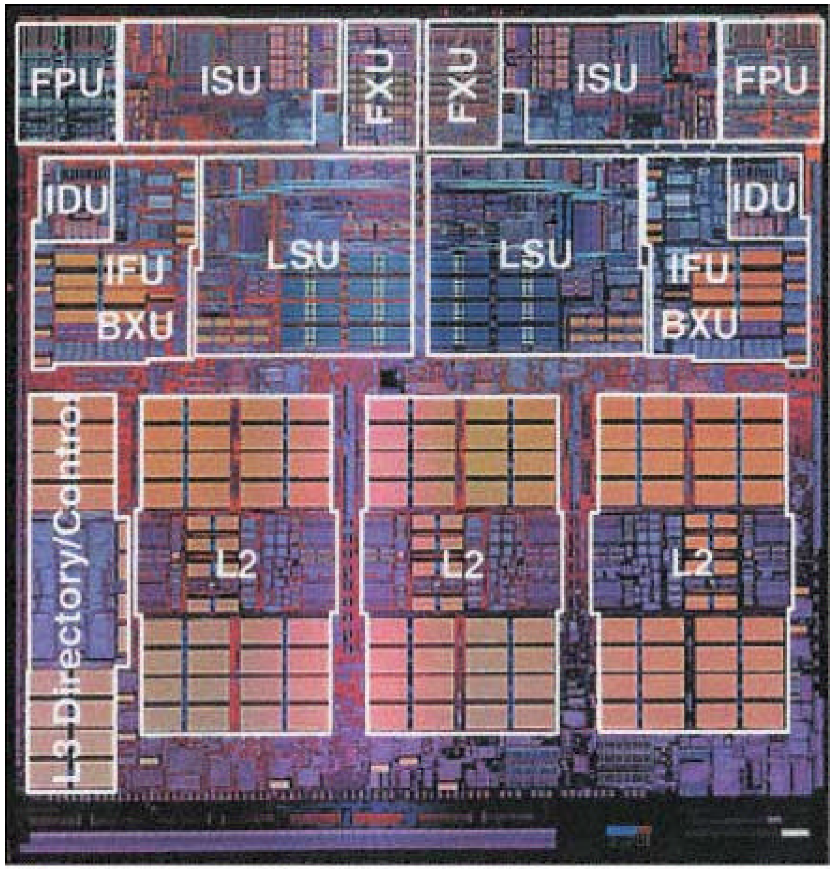
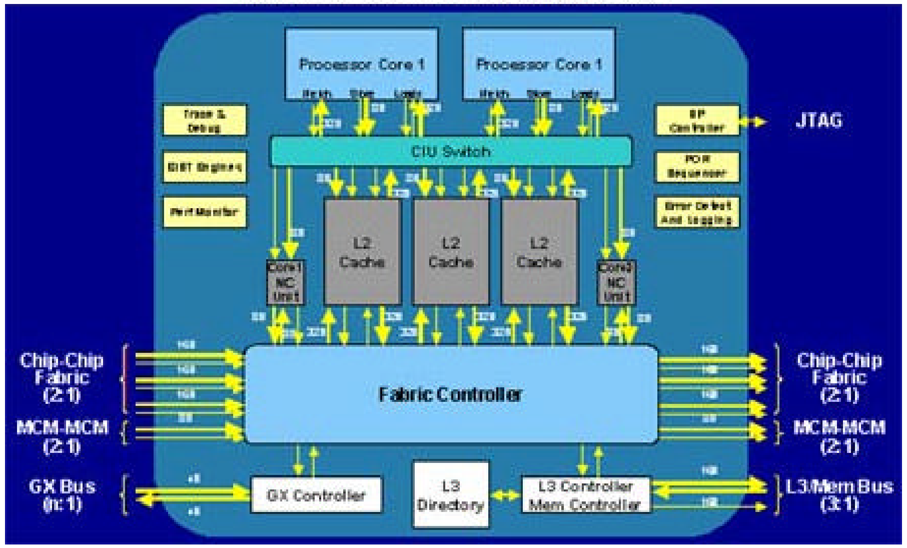


Figure 2

POWER4 chip photograph showing the principal functional units in the microprocessor core and in the memory subsystem.

Figure 1: POWER4 Chip Logical View

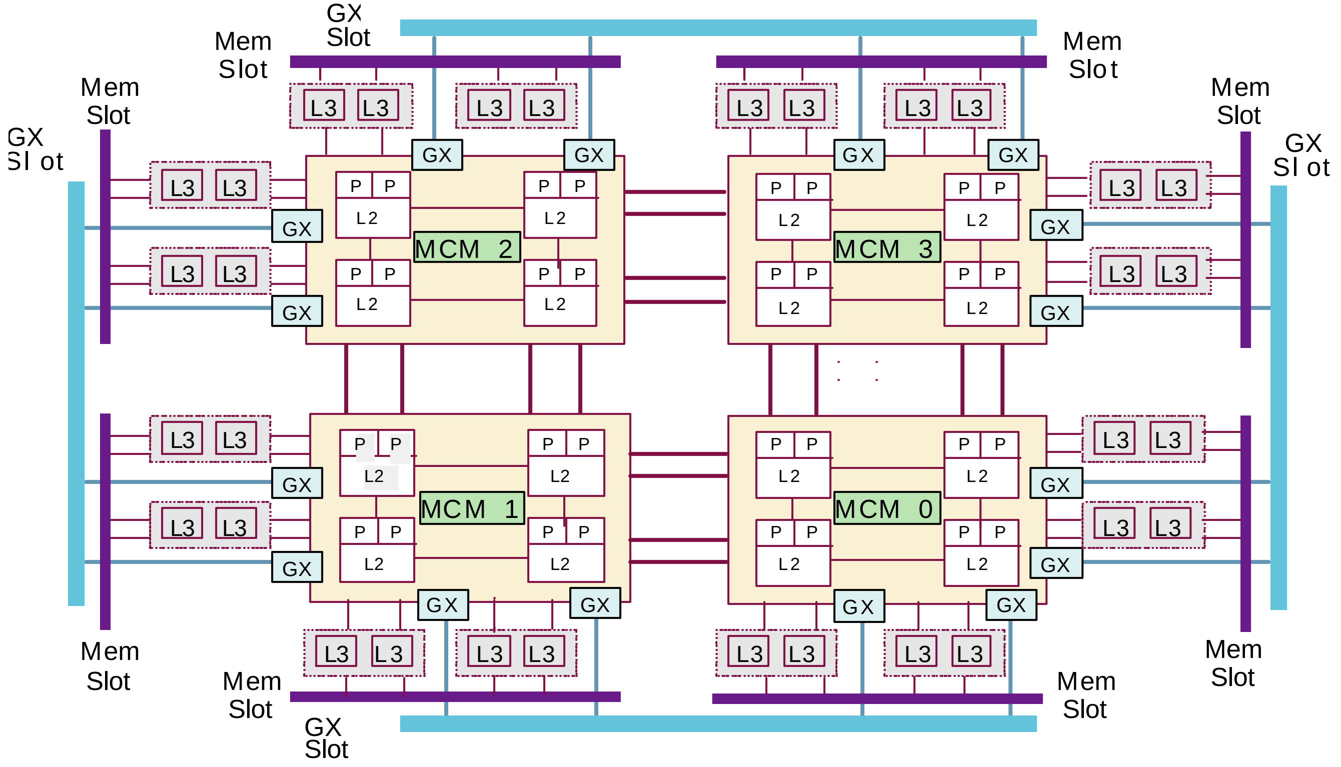


http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/power4.html

J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy,
" POWER4 system microarchitecture",
IBM Journal of Research and Development, Vol46, No. 1, 2002

IBM pSeries690 RegattaH

- Up to 16 Power4: 32 way SMP Server
  - L1(D) : 64 KB (32KB/processor, 2 way assoc.), L1(I): 128 KB (64KB/processor, Direct map)
  - L2 : 1.5 MB (shared cache with 2 procs., 4~8 way assoc.)
  - L3 : 32 MB (external, 8 way assoc.) [x 8 = 256 MB]



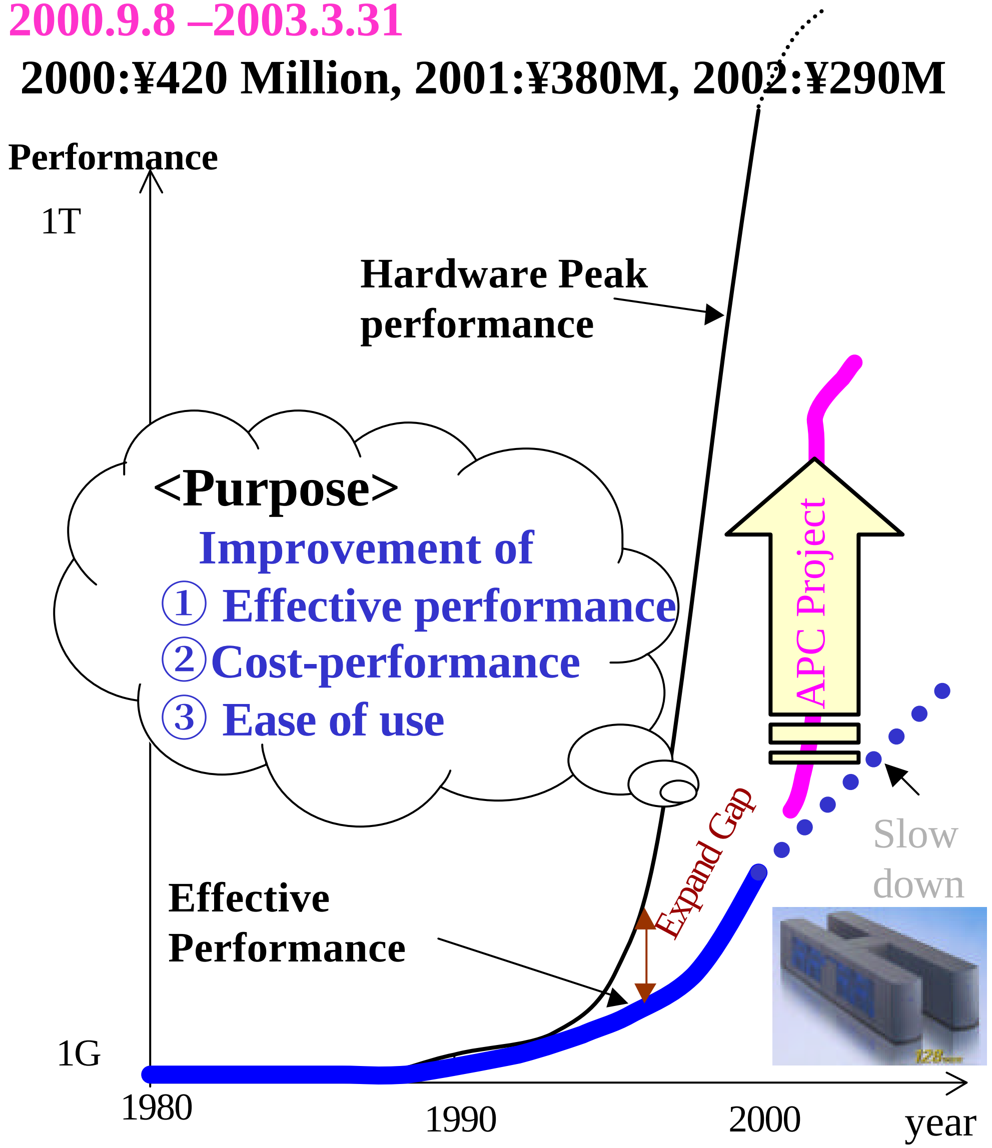
Four 8-way MCM Features Assembled into a 32-way pSeries 690

http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/p690_config.html

Advanced Parallelizing Compiler Technology Project

2000.9.8 –2003.3.31

2000:¥420 Million, 2001:¥380M, 2002:¥290M



Theoretical maximum performance vs. Effective performance of HPC

Background and Problems

①Adoption of parallel processing as a core technology on PC to HPC

② Increase of importance of software on IT

③ Need for improvement of cost-performance and usability

Contents of Research and Development

① R & D of advanced parallelizing compiler
Multigrain, Data localization, Overhead hiding

② R & D of Performance evaluation technology for parallelizing compilers

Goal: Double the effective performance

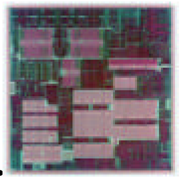
Ripple Effect

① Development of competitive next generation PC and HPC

② Putting the innovative automatic parallelizing compiler technology to practical use

③ Development and market acquisition of future single-chip multiprocessors

④ Boosting R&D in the following many fields:
IT, Bio-tech., Device, Earth environment, Next-generation VLSI design, Financial engineering, Weather forecast, New clean energy, Space development, Automobile, Electric Commerce, etc



Organization of Advanced Parallelizing Compiler Project (2)

JIPDEC (Japan Information Processing Development Corporation) *(administrative corporation: Contractor of APC Project)*

APC Propulsive Section

Project Leader : Prof. Hironori KASAHARA (Waseda Univ.)
Sub-Leaders : Assoc. Prof. Hayato YAMANA (Waseda Univ.)
Dr. Hanpei KOIKE (AIST)

Central Laboratory (APC Lab.) – Waseda Univ.

Int'l Advisory Board

- Prof. David A. Padua
- Prof. Monica S. Lam
- Prof. Rudolf Eigenmann
- Prof. Francois Irigoin

Group of Advanced Parallelizing Compiler Technology

Group Leader: Koh HOTTA (Fujitsu)

Fujitsu Ltd.(*)
Hitachi Ltd.(*)
AIST(**) (National Institute of Advanced Industrial Science and Technology)
Waseda Univ.(**)
Toho Univ.(***)

Group of Performance Evaluatiojn for Parallelizing Compiler

Group Leader: Tokuro ANZAKI (Hitachi)

Fujitsu Ltd.(*)
Hitachi Ltd.(*)
Tokyo Institute of Technology (***)
Univ. of Electro-Communications(***)
Waseda Univ.(**)

(*) Researchers of Fujitsu and Hitachi are on loan to JIPDEC
(**) Joint Research contractors (**) Subcontractors

Research Parallelizing Compilers

- **Automatic loop parallelization**

  <Data Dependence Analysis & Restructuring>

  - **Runtime dependence analysis, Symbolic analysis, Interprocedure analysis, Unimoduler transformation, Array prizatization, Fusion, Unrolling etc.**

➡ **Polaris, SUIF , Promis (Parafrase2) …**

  - **Limitation of loop parallelism**
    - **Sequential loops and parts of program except loops**
    - **Fewer loop iterations than the number of processors**

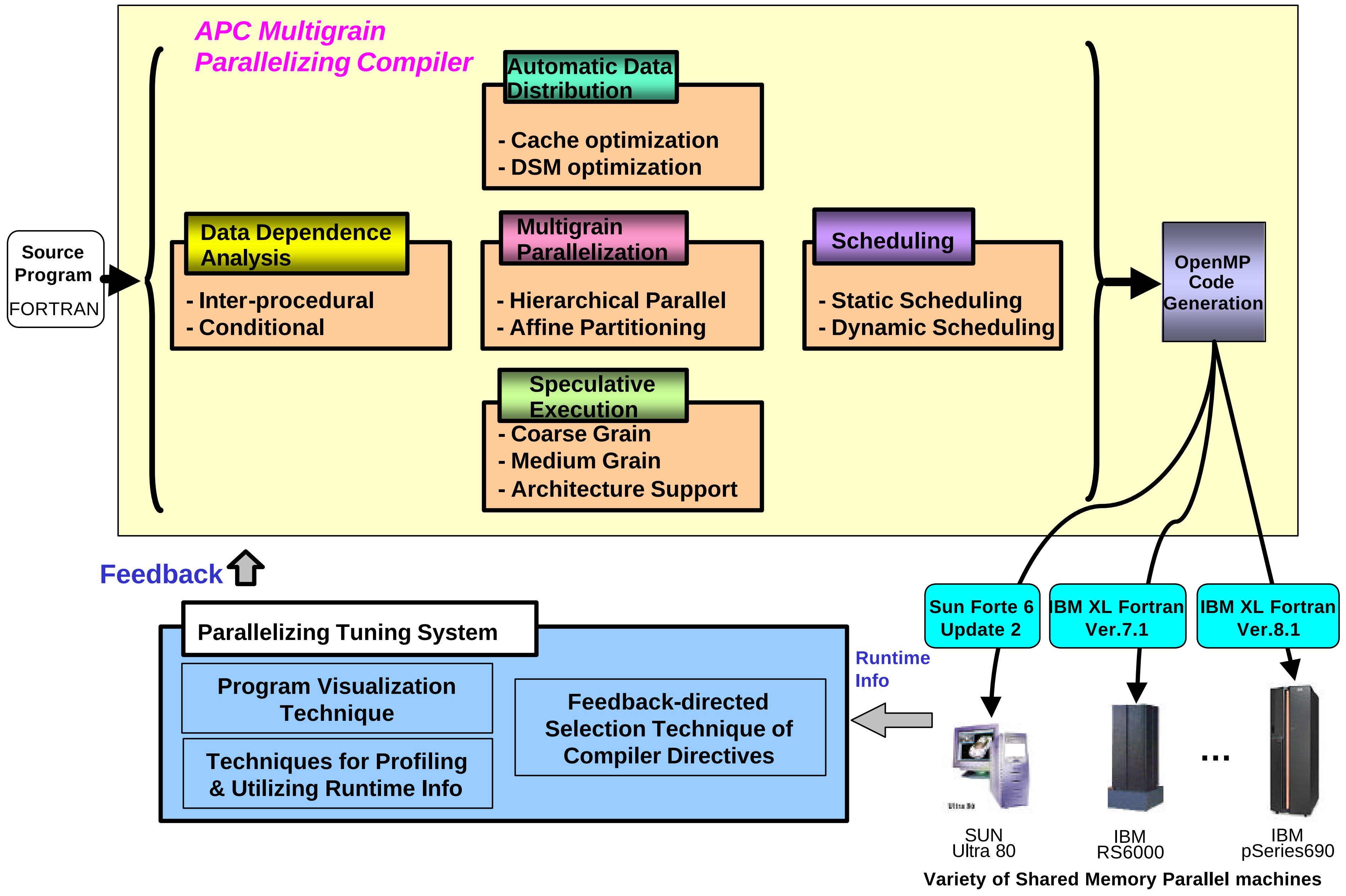
Multigrain Parallel Processing in OSCAR parallelizing compiler

- **Coarse grain task parallelism**
  - **Among subroutines, loops & basic blocks**
  - **Statically or dynamically schedule to processors or PCs (Processor Clusters) at compile or run time**
- **Loop parallelism**
  - **Among loop iterations**
  - **Schedule to processors or PCs**
- **(Near) Fine grain parallelism**
  - **Among statements in a basic block**
  - **Statically schedule to processors in a PC**
- **Their hierarchical combination**

Research Topics in Advanced Parallelizing Compiler Project

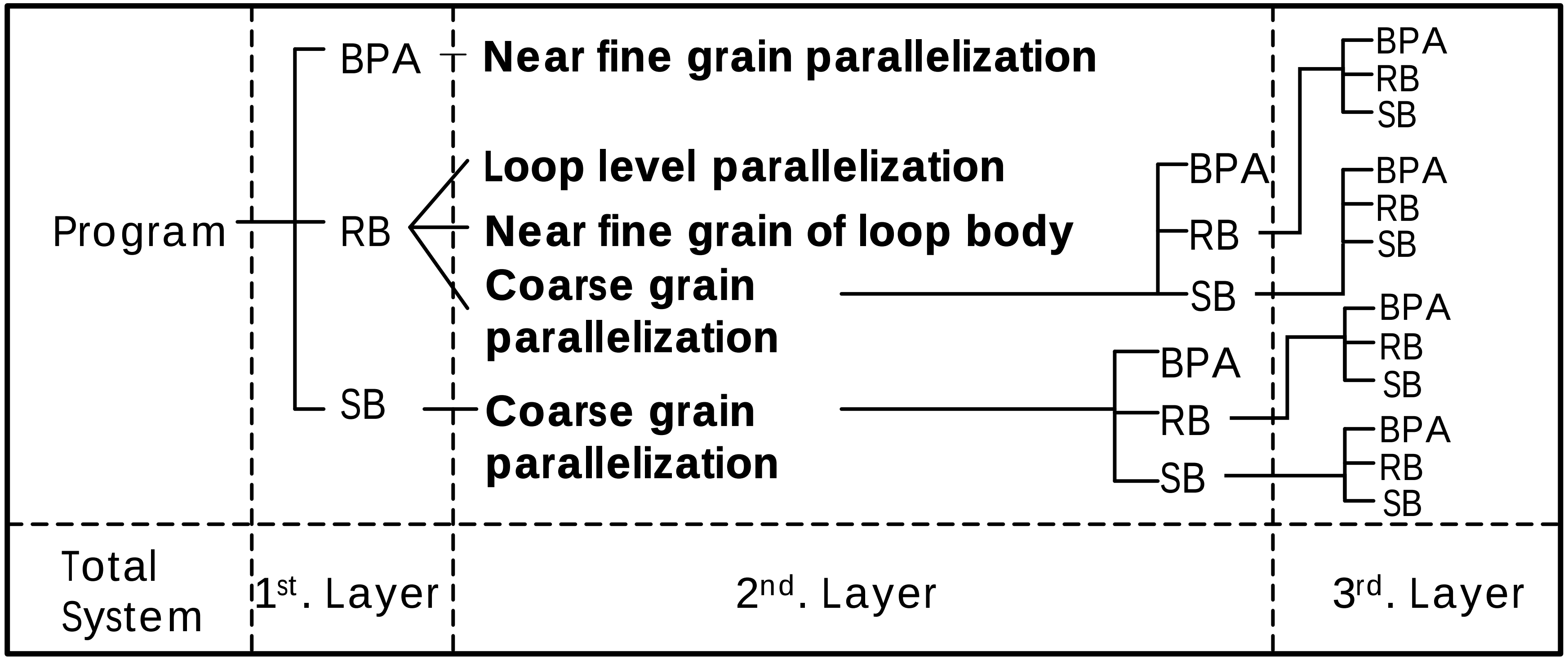
- Multigrain parallelization technology
  - Data dependence analysis (Inter-procedural, Runtime)
  - Automatic selection of suitable grain
  - Automatic data distribution (DSM, Cache, Local Mem.)
  - Scheduling (Load balancing, Data transfer overhead minimization)
  - Speculative execution
  - Tuning tools
  - Use of OpenMP+ as an intermediate language
- Performance evaluation of compiler
  - Evaluation of individual parallelization technology
  - Evaluation of total compiler performance
    - Selection of benchmark programs which can clearly show performance of compiler

APC Compiler Organization



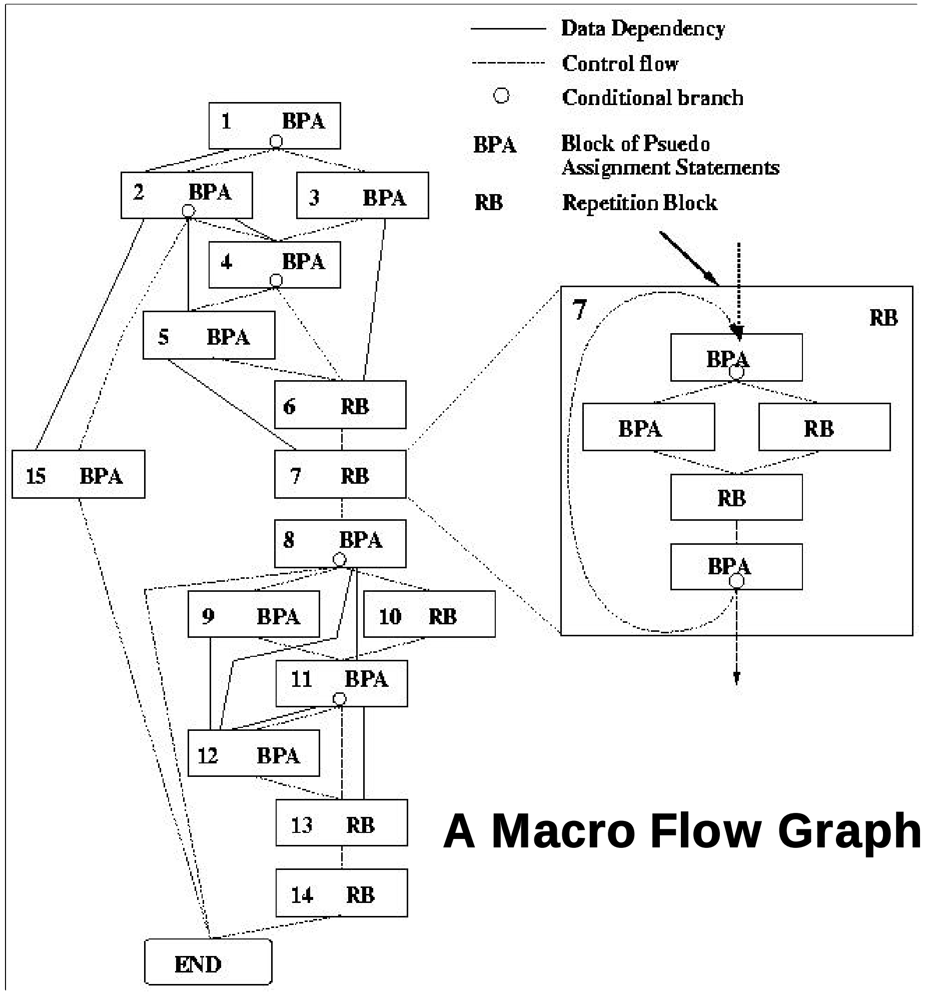
Generation of Coarse Grain Tasks

- **Macro-tasks (MTs)**
  - Block of Pseudo Assignments (**BPA**): Basic Block (BB)
  - Repetition Block (**RB**) : outermost natural loop
  - Subroutine Block (**SB**): subroutine

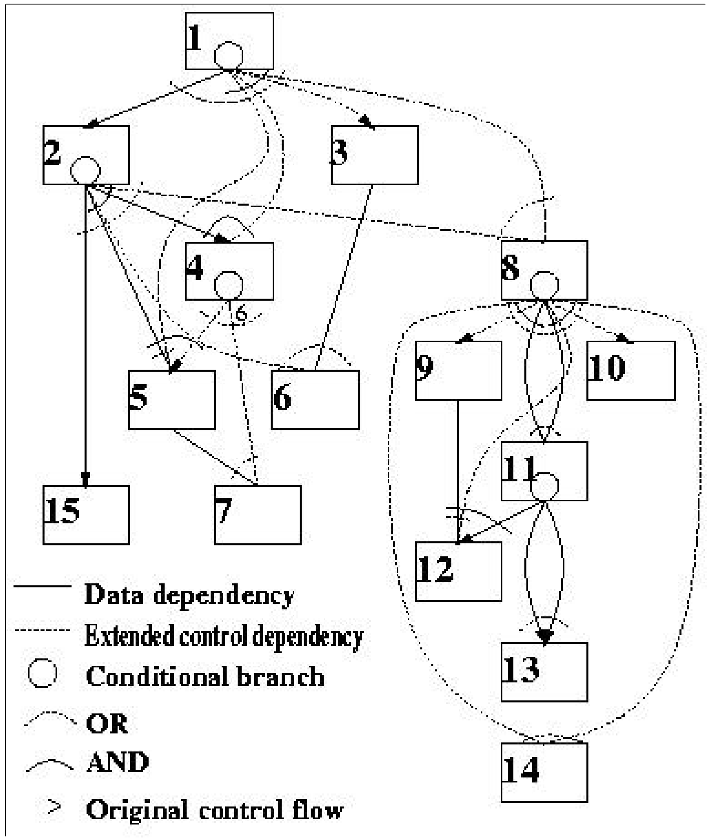


Program
BPA
RB
SB
Near fine grain parallelization
Loop level parallelization
Near fine grain of loop body
Coarse grain parallelization
Coarse grain parallelization
BPA
RB
SB
BPA
RB
SB
BPA
RB
SB
BPA
RB
SB
BPA
RB
SB
BPA
RB
SB
Total System
1st. Layer
2nd. Layer
3rd. Layer

Earliest Executable Condition Analysis for Coarse Grain Tasks (Macro-tasks)



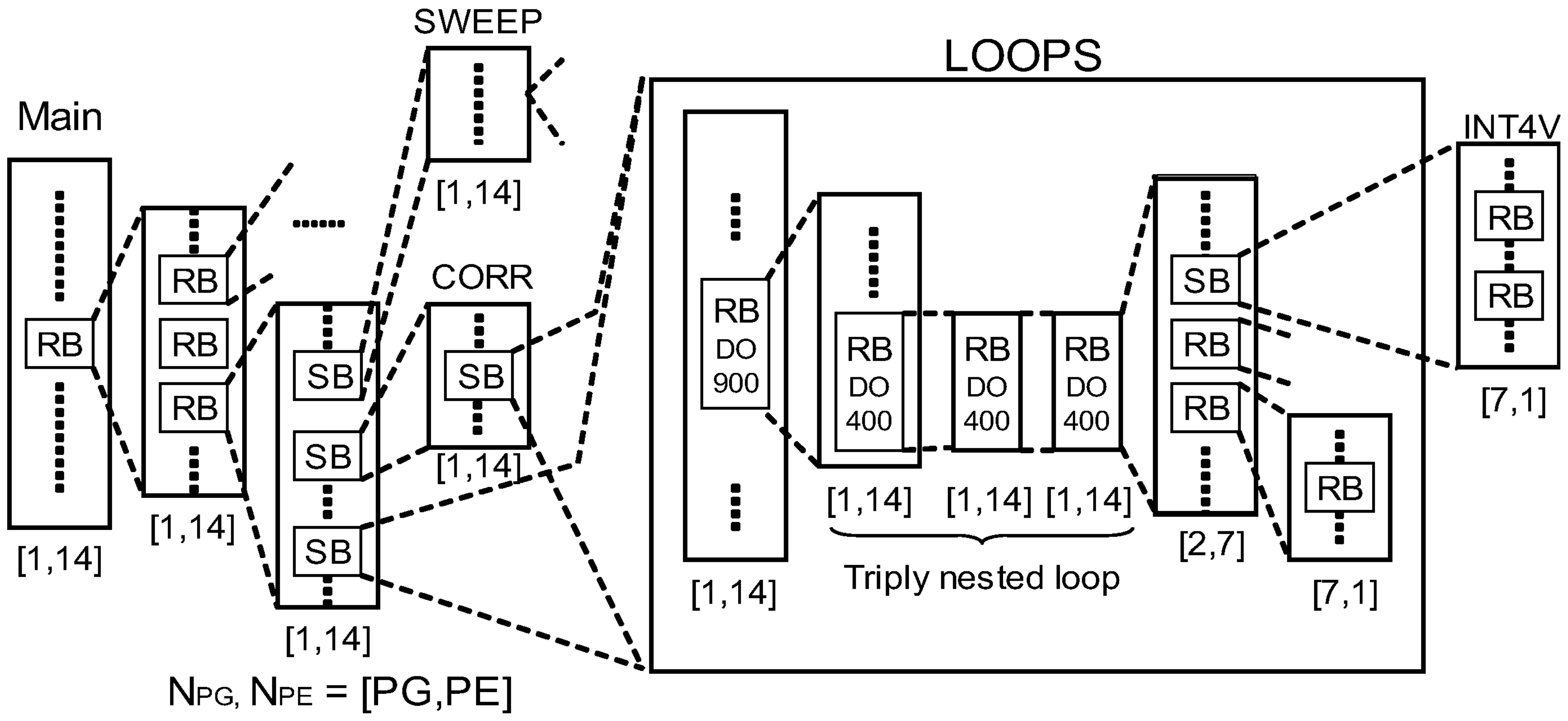
A Macro Flow Graph



A Macro Task Graph

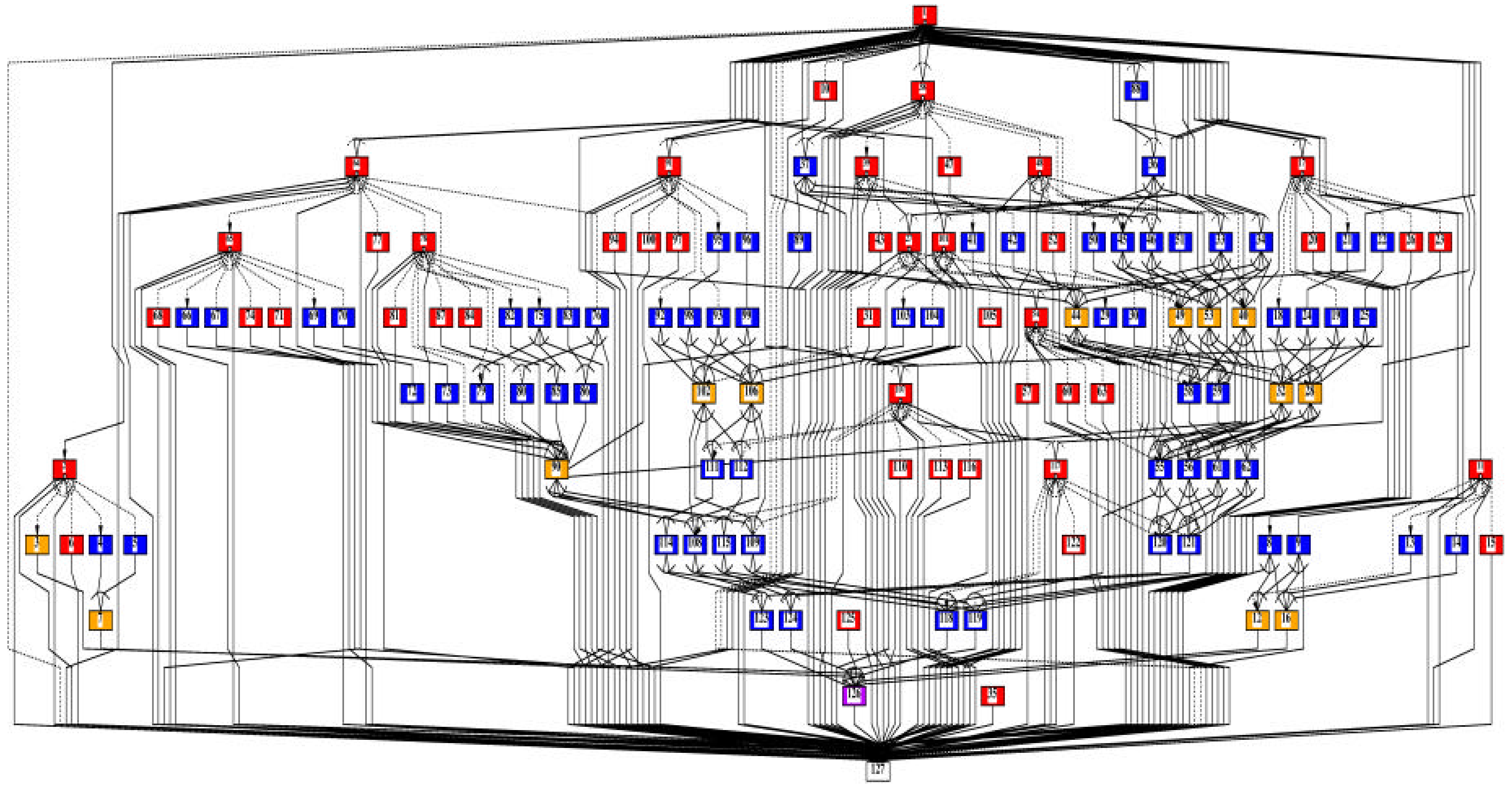
Automatic processor assignment in 103.su2cor

- **Using 14 processors**
  - **Coarse grain parallelization within DO400 of subroutine LOOPS**



MTG of Su2cor-LOOPS-DO400

- Coarse grain parallelism PARA_ALD = 4.3

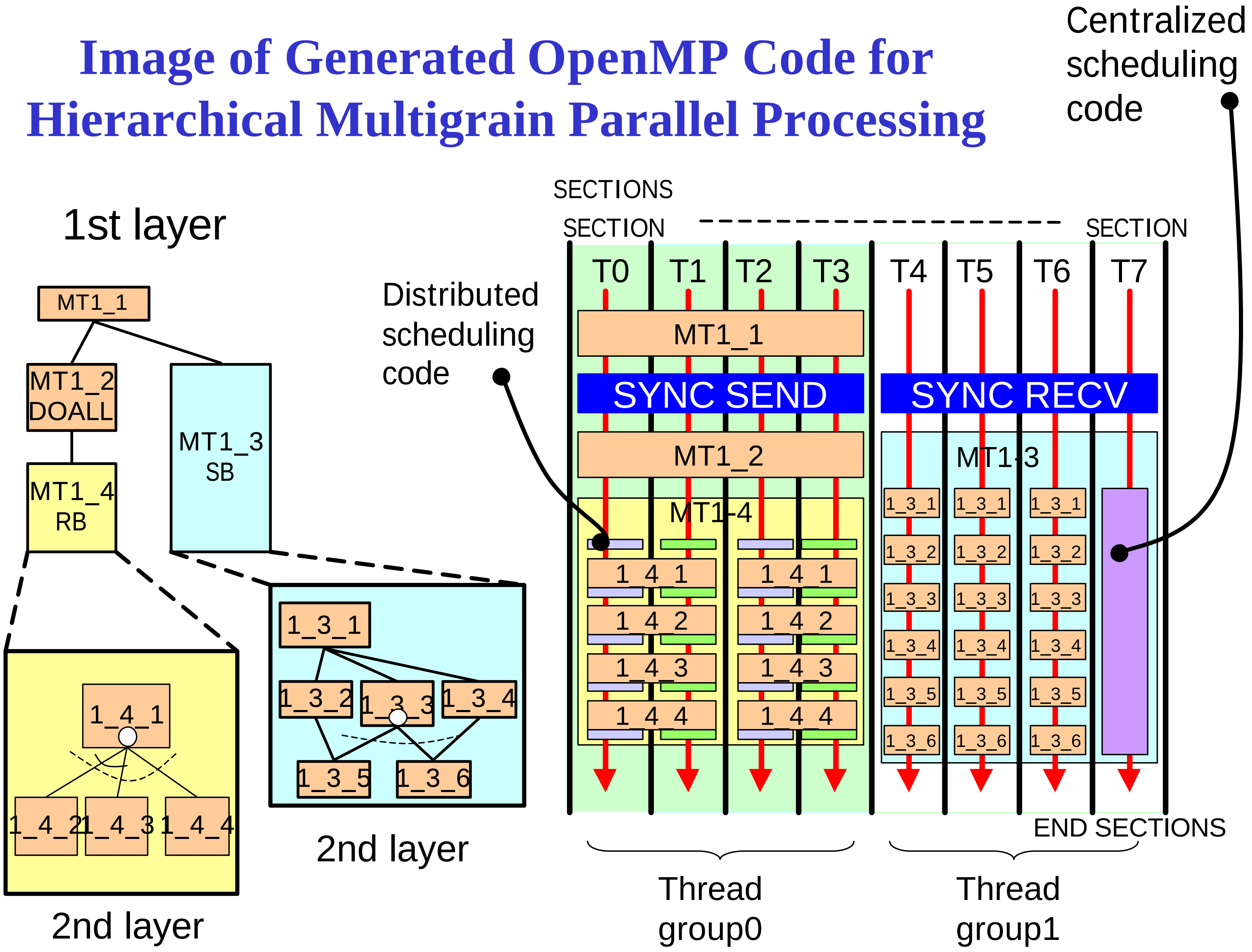


🟦 DOALL 🟪 Sequential LOOP 🟨 SB 🟥 BB

Code Generation Using OpenMP

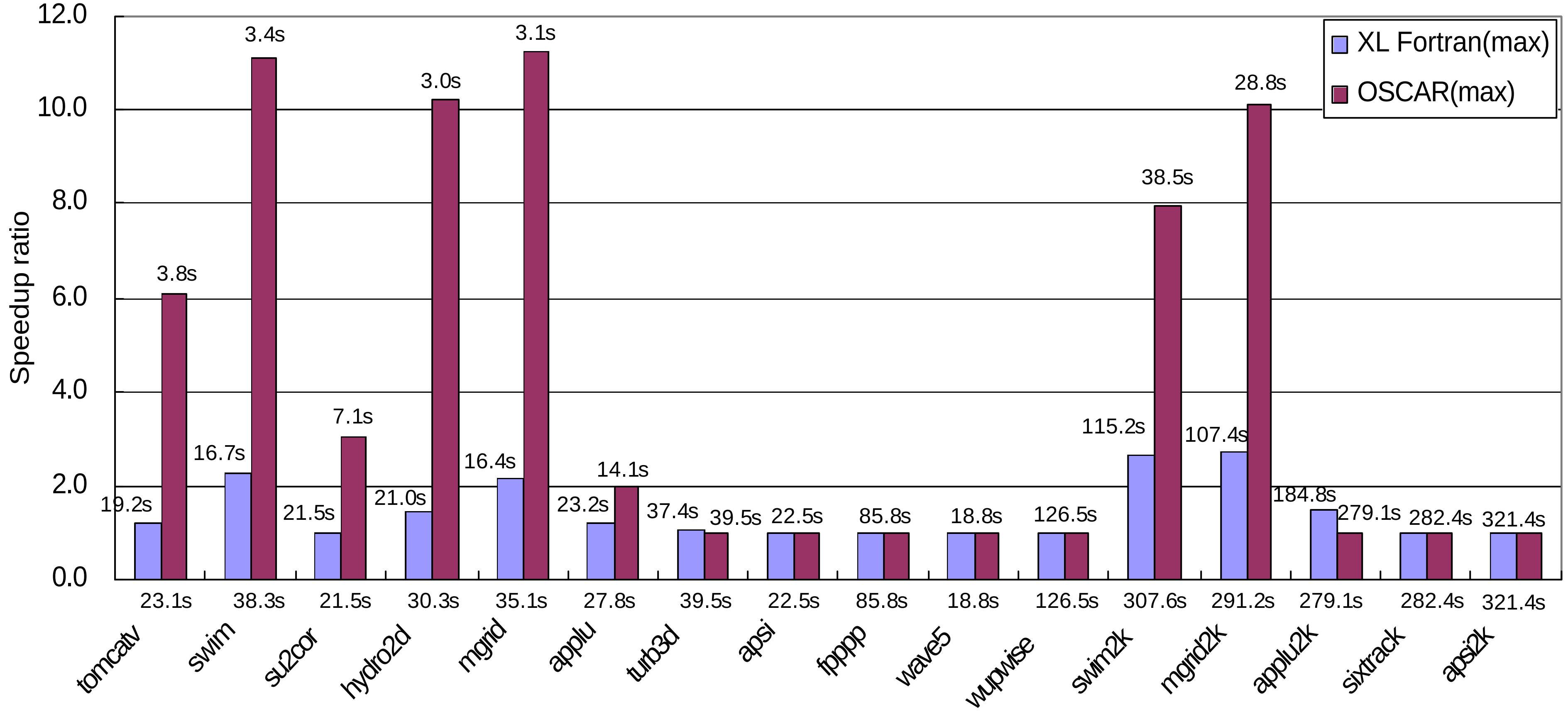
- Compiler generates a parallelized program using OpenMP API
- One time single level thread generation
  - Threads are forked only once at the beginning of a program by OpenMP “PARALLEL SECTIONS” directive
  - Forked threads join only once at the end of program
- Compiler generates codes for each threads using static or dynamic scheduling schemes
- Extension of OpenMP for hierarchical processing is not required

Image of Generated OpenMP Code for Hierarchical Multigrain Parallel Processing

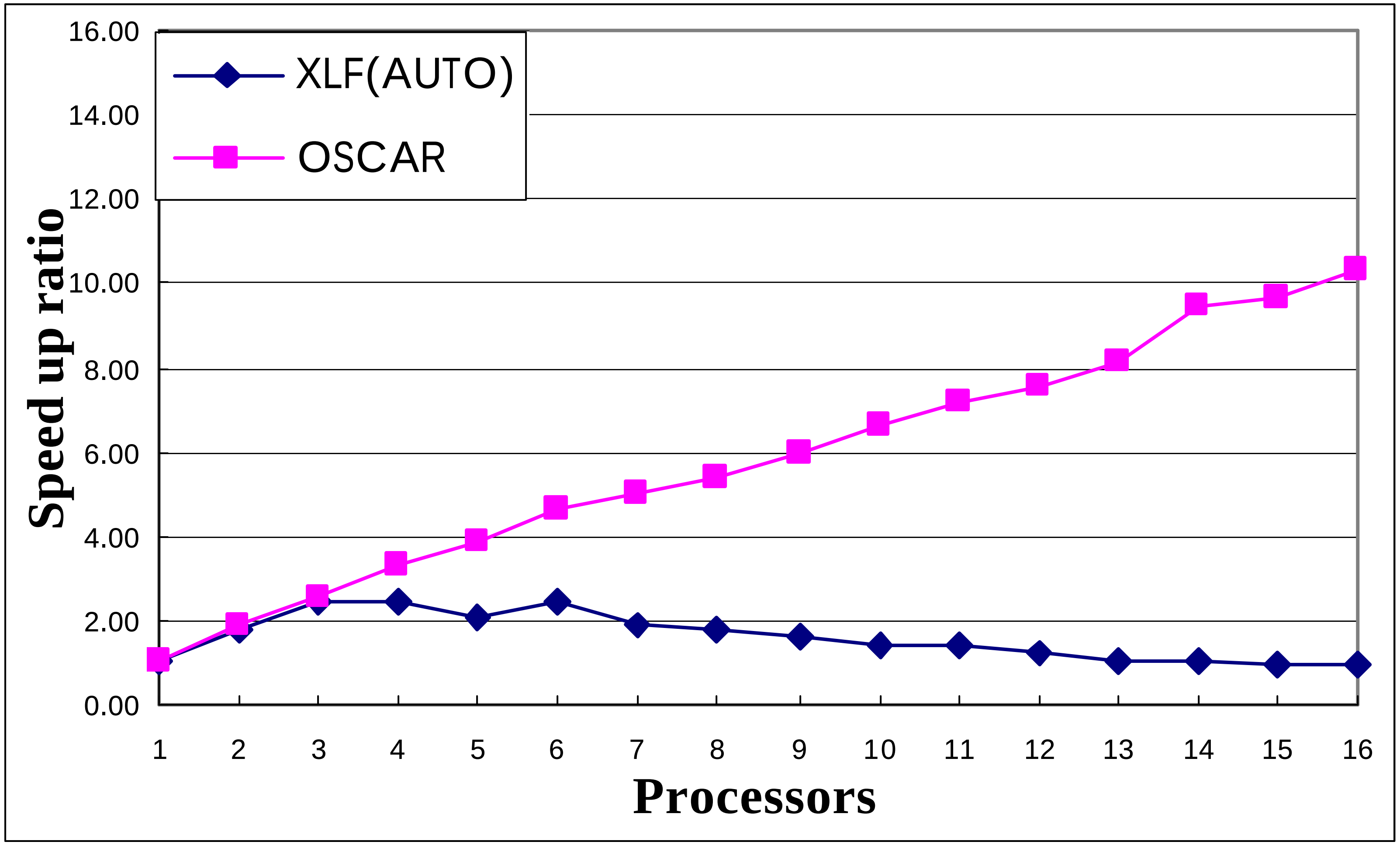


Performance of Multigrain Parallelization on 16 processor High-end Server IBM pSeries690

- IBM XL Fortran for AIX Version 8.1
  - Sequential execution : -O5 -qarch=pwr4
  - Automatic loop parallelization : -O5 -qsmp=auto -qarch=pwr4
  - OSCAR compiler : -O5 -qsmp=noauto -qarch=pwr4 (su2cor: -O4 -qstrict)

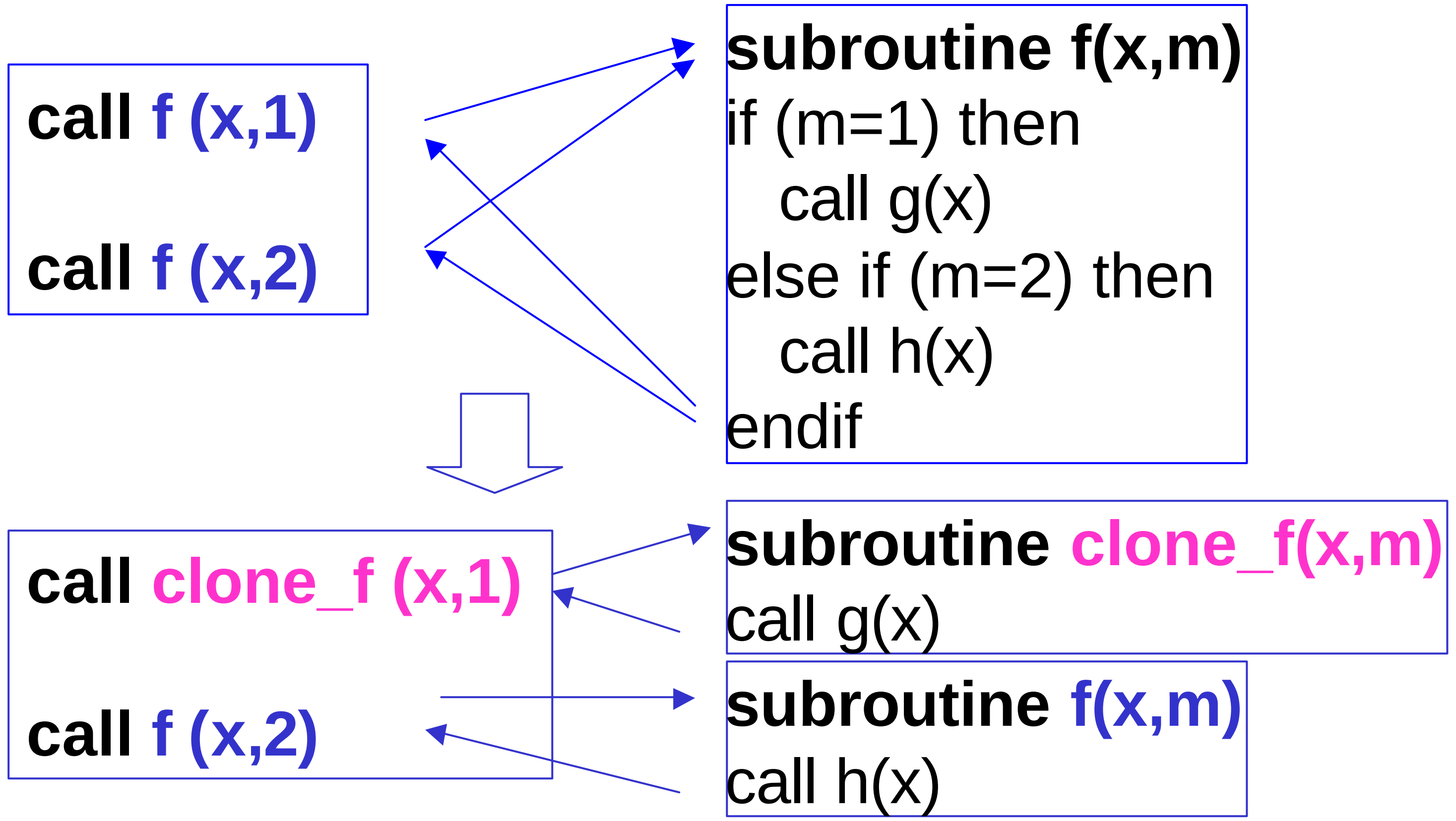


Performance of Multigrain Parallel Processing for 102.swim on IBM pSeries690



Aggressive Constant Propagation

It makes clone procedures,
propagates constants beyond procedure boundaries,
and evaluates expressions at compile time.



```
call f (x,1)

call f (x,2)
```

```
subroutine f(x,m)
if (m=1) then
   call g(x)
else if (m=2) then
   call h(x)
endif
```

```
call clone_f (x,1)

call f (x,2)
```

```
subroutine clone_f(x,m)
call g(x)
```

```
subroutine f(x,m)
call h(x)
```

Interprocedural Parallelization

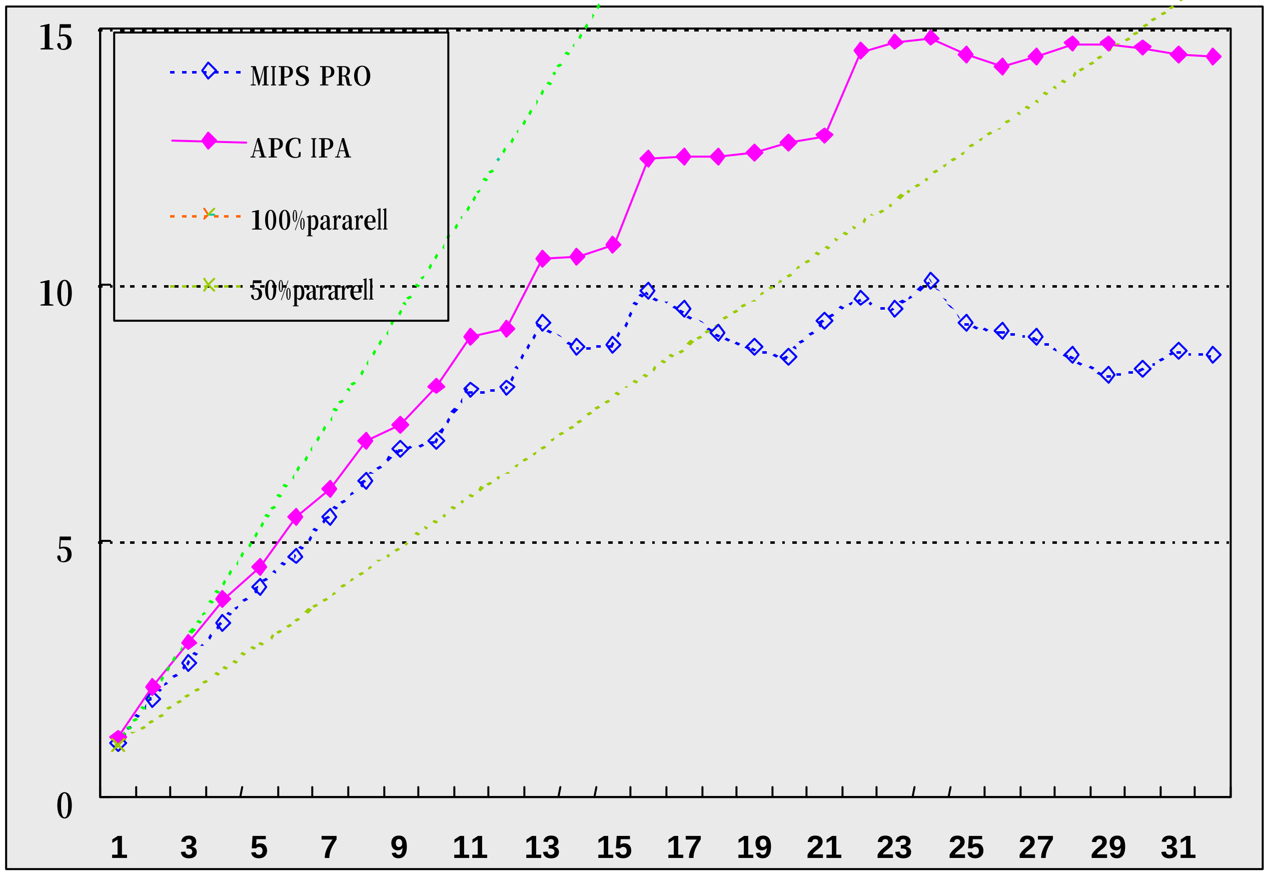
It can parallelize the loops including procedure calls and enclose contiguous parallel loops in one parallel region.

```
do i=1, n
   ...
  call f(x,y)
    ...
end do
do j =1,n
  ...
end do
```

```
!$OMP PARALLEL
    !$OMP DO
      do i=1, n
        call f(x, y)
      enddo
    !$OMP END DO NOWAIT
    !$OMP DO
      do j=1, n
          ...
      enddo
    !$OMP END DO
!$OMP END PARALLEL
```

Performance of InterproceduralAnalysis

SPEC CFP95 turb3d on Origin 2000



- 🟪 Interprocedural Analysis in APC compiler
- 🟦 MIPSPro Fortran V.7.30

Affine Partitioning [Lim & Lam97]

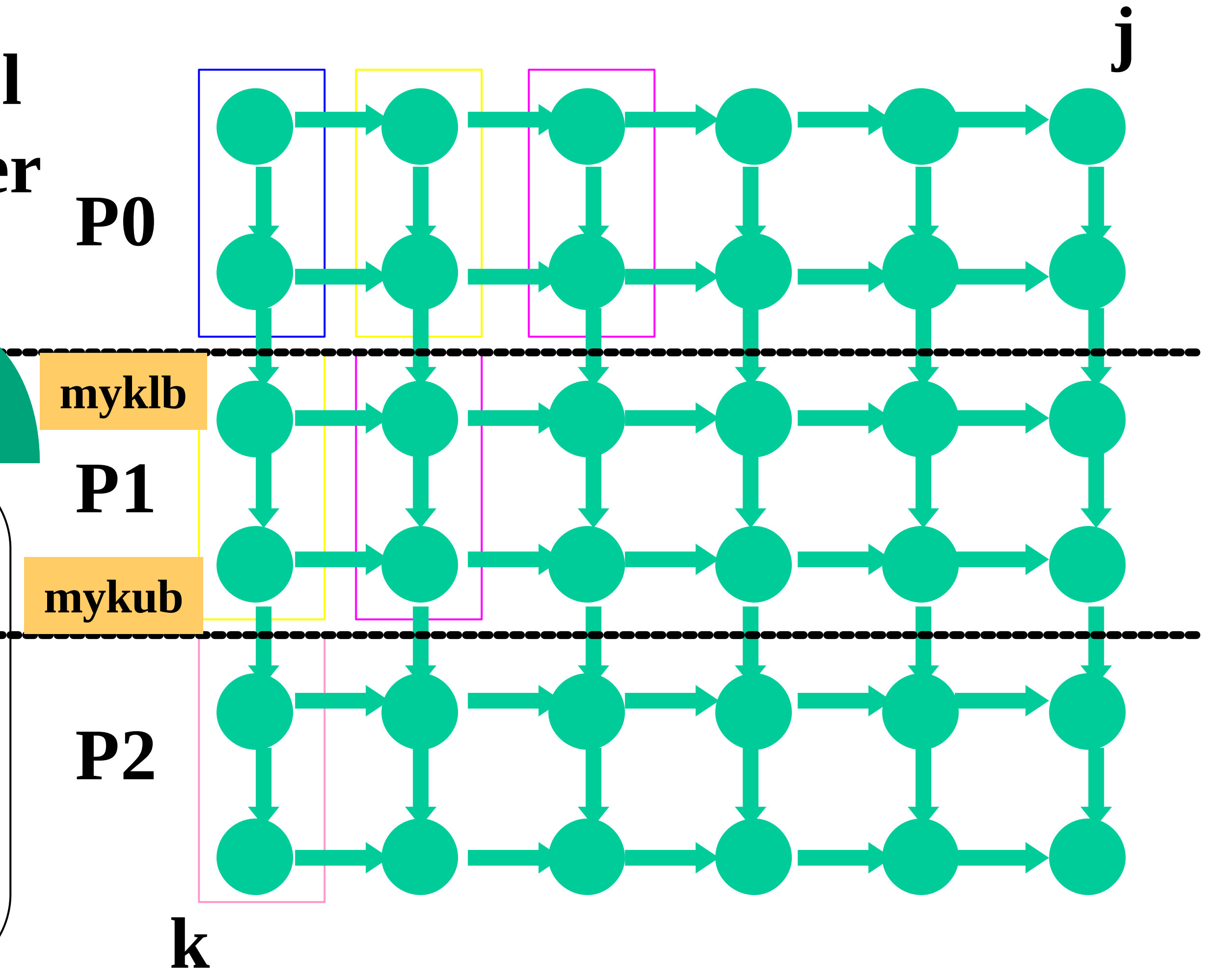
- The followings are considered at the same time
  - parallelization
  - improve data locality
  - reduce synchronization overhead

- A lot of transformations can be done automatically

- Extract pipeline parallelism

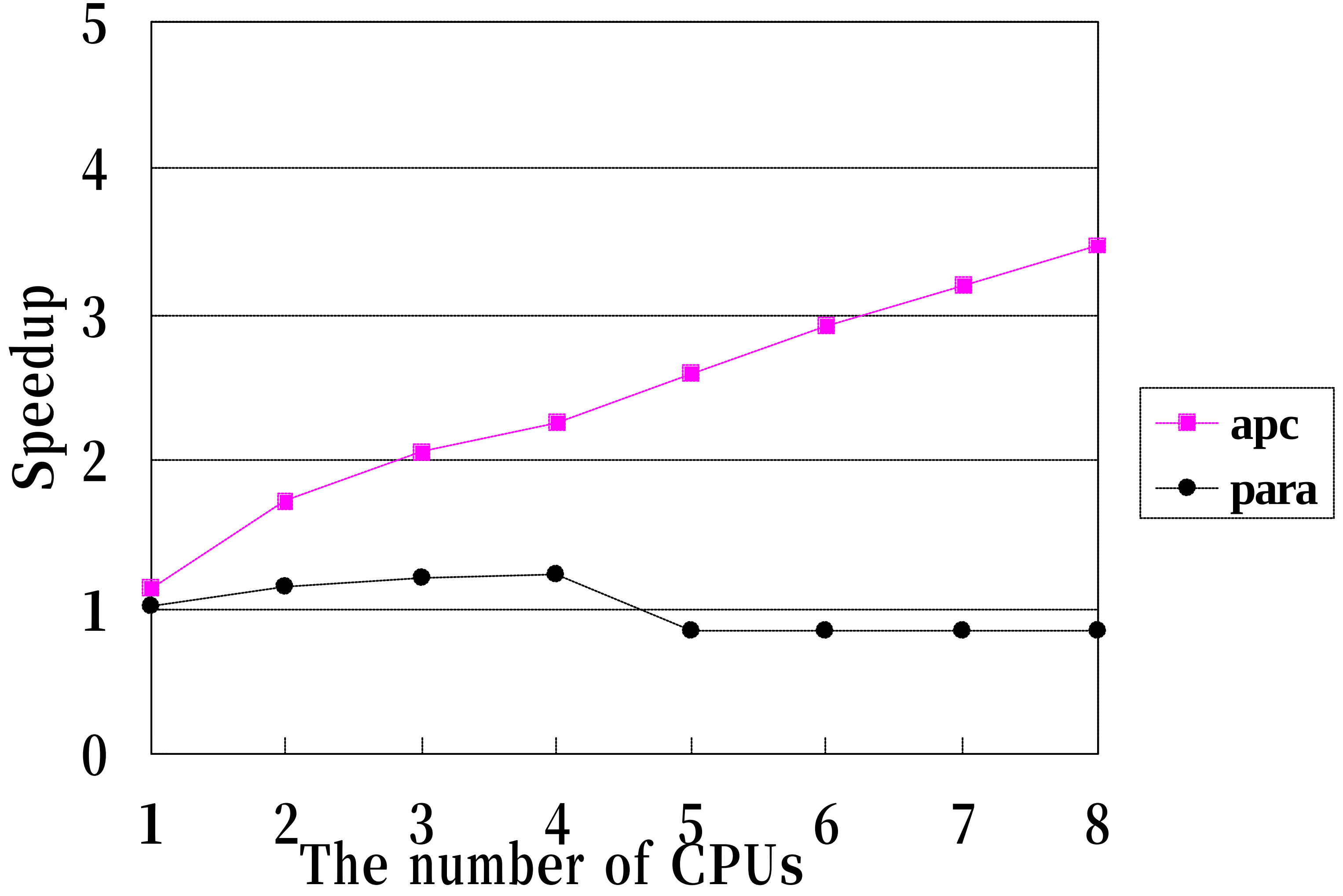
Generated pipelined code in OpenMP

Consider parallel execution order



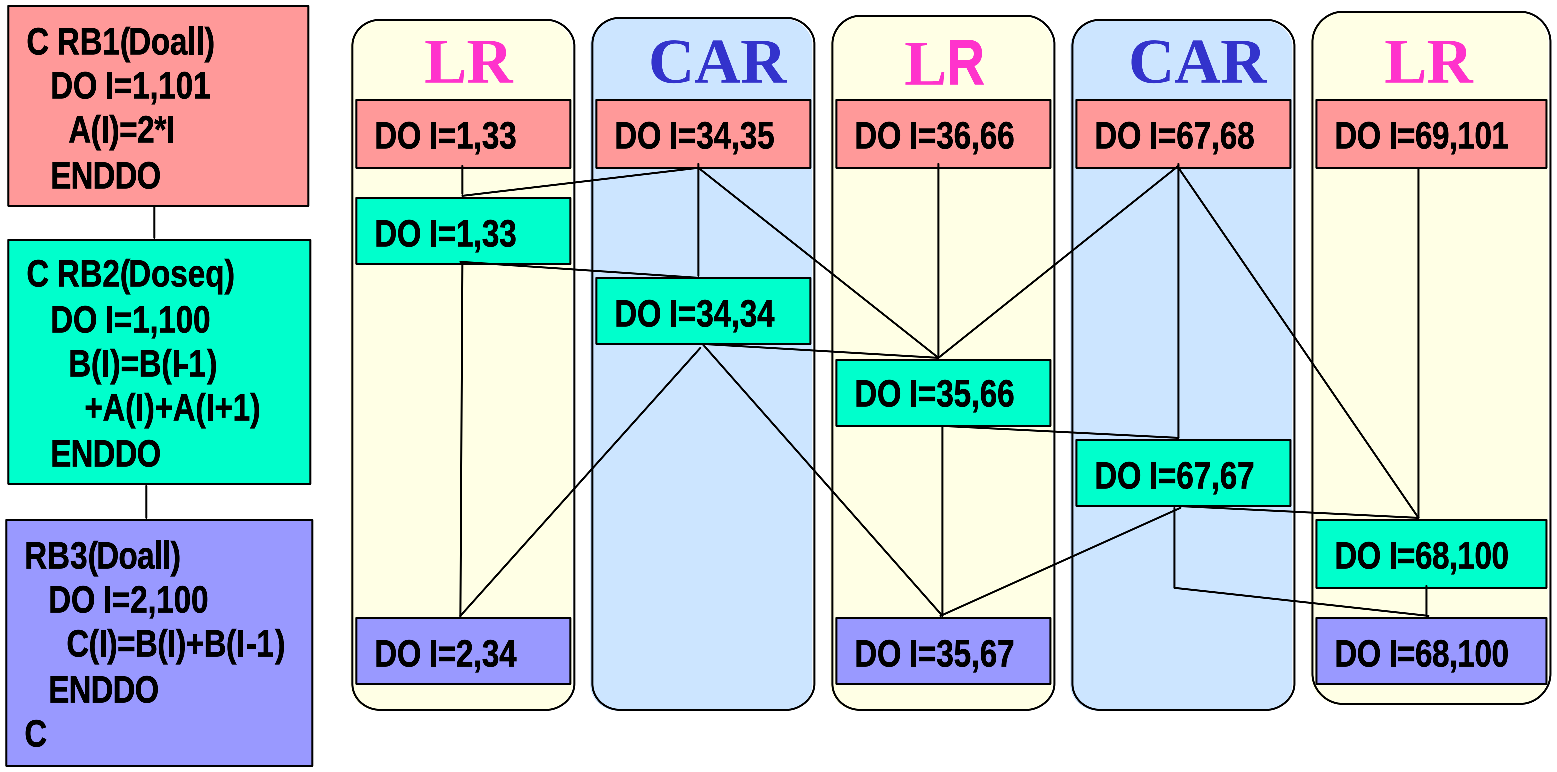
```
do  j = 1, ny-2
     waitPrev()
    do  k = myklb, mykub
           ...
           ...
    enddo
     signalToNext()
enddo
```

Speedup of applu on HP Alpha Server (8 Processors) by Affine Partitioning



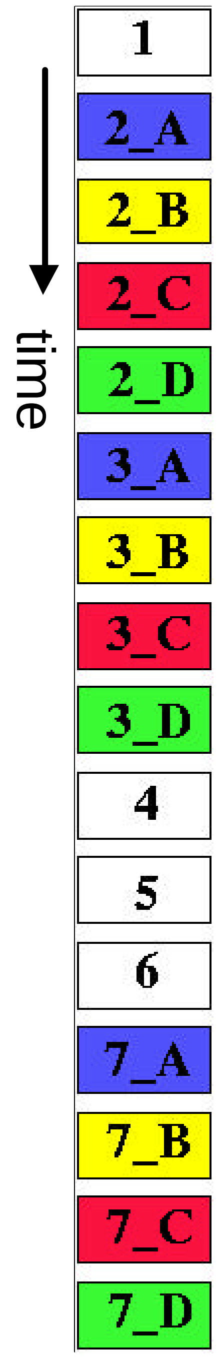
Data-Localization
Loop Aligned Decomposition

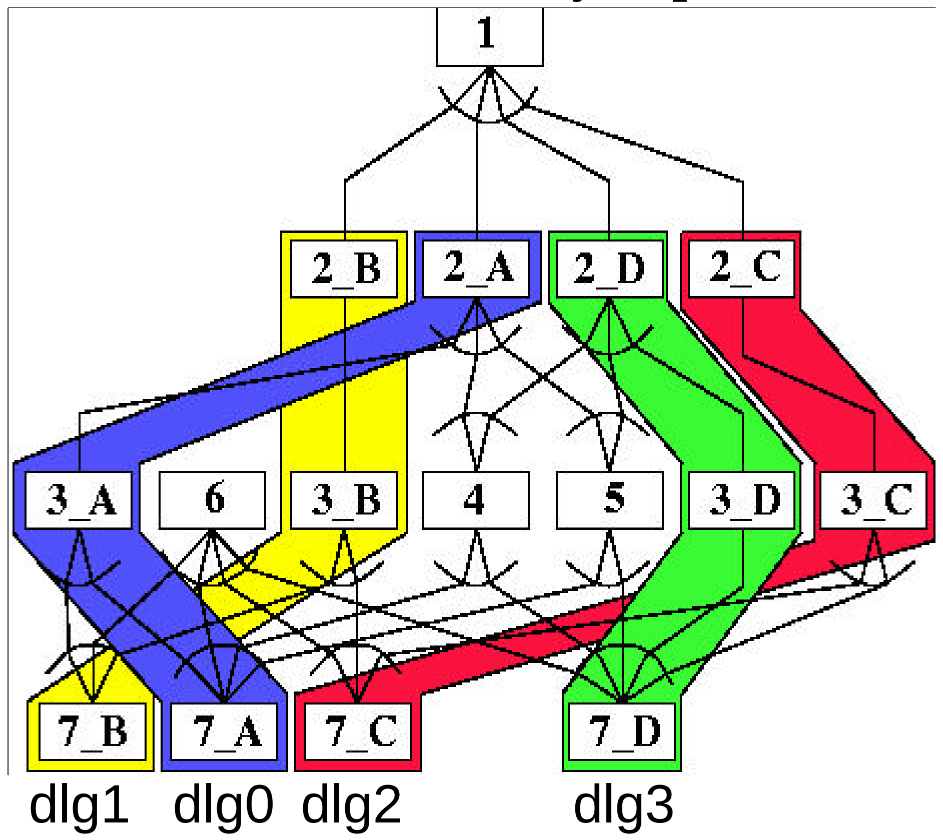
- Decompose multiple loop (Doall and Seq) into CARs and LRs considering inter-loop data dependence.
  - Most data in LR can be passed through LM.
  - LR: Localizable Region, CAR: Commonly Accessed Region

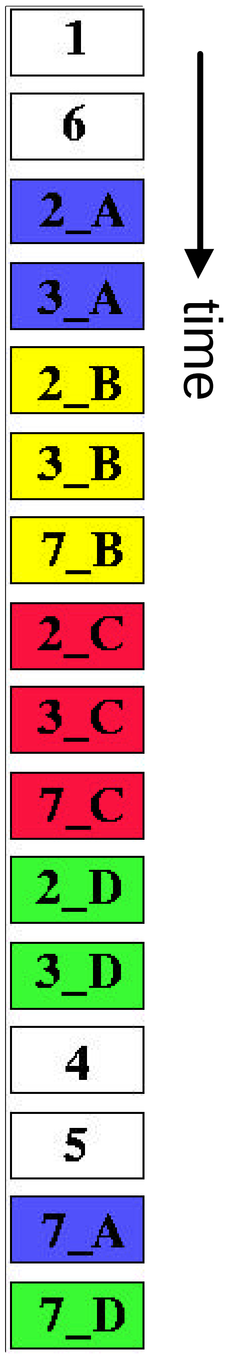


Partial Static Assignment

Dynamic scheduler assigns macro-tasks in a DLG to the same processors as consecutively as possible







Original execution order on single processor

Dynamic scheduling result on single processor

Effective cache usage

Data Layout for Removing Line Conflict Misses by Array Dimension Padding

Declaration part of arrays in spec95 swim

before padding

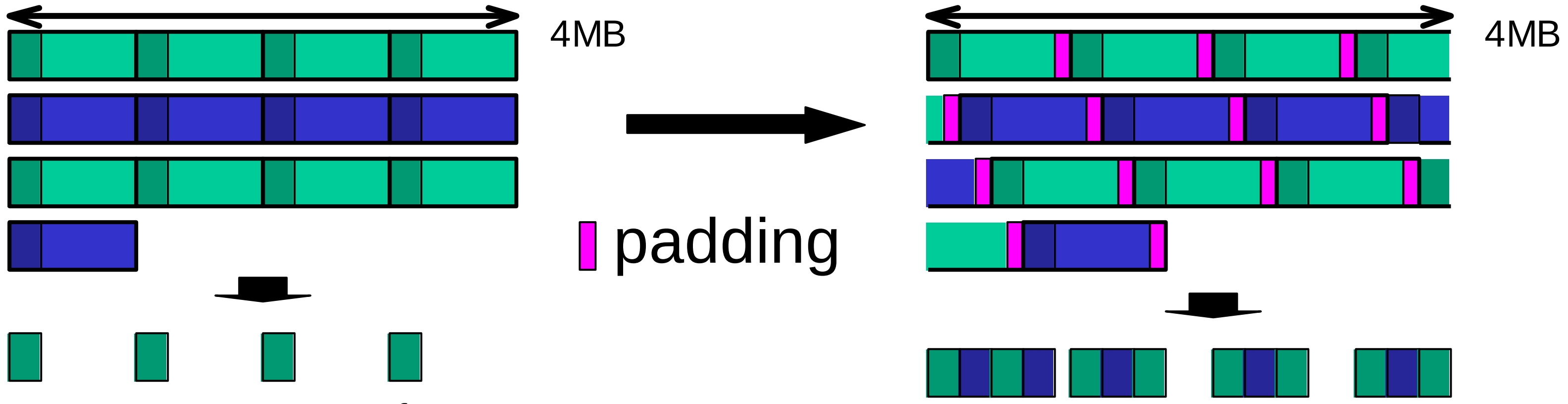
```
 PARAMETER (N1=513, N2=513)

 COMMON  U(N1,N2), V(N1,N2), P(N1,N2),
*        UNEW(N1,N2), VNEW(N1,N2),
1        PNEW(N1,N2), UOLD(N1,N2),
*        VOLD(N1,N2), POLD(N1,N2),
2        CU(N1,N2), CV(N1,N2),
*        Z(N1,N2), H(N1,N2)
```

after padding

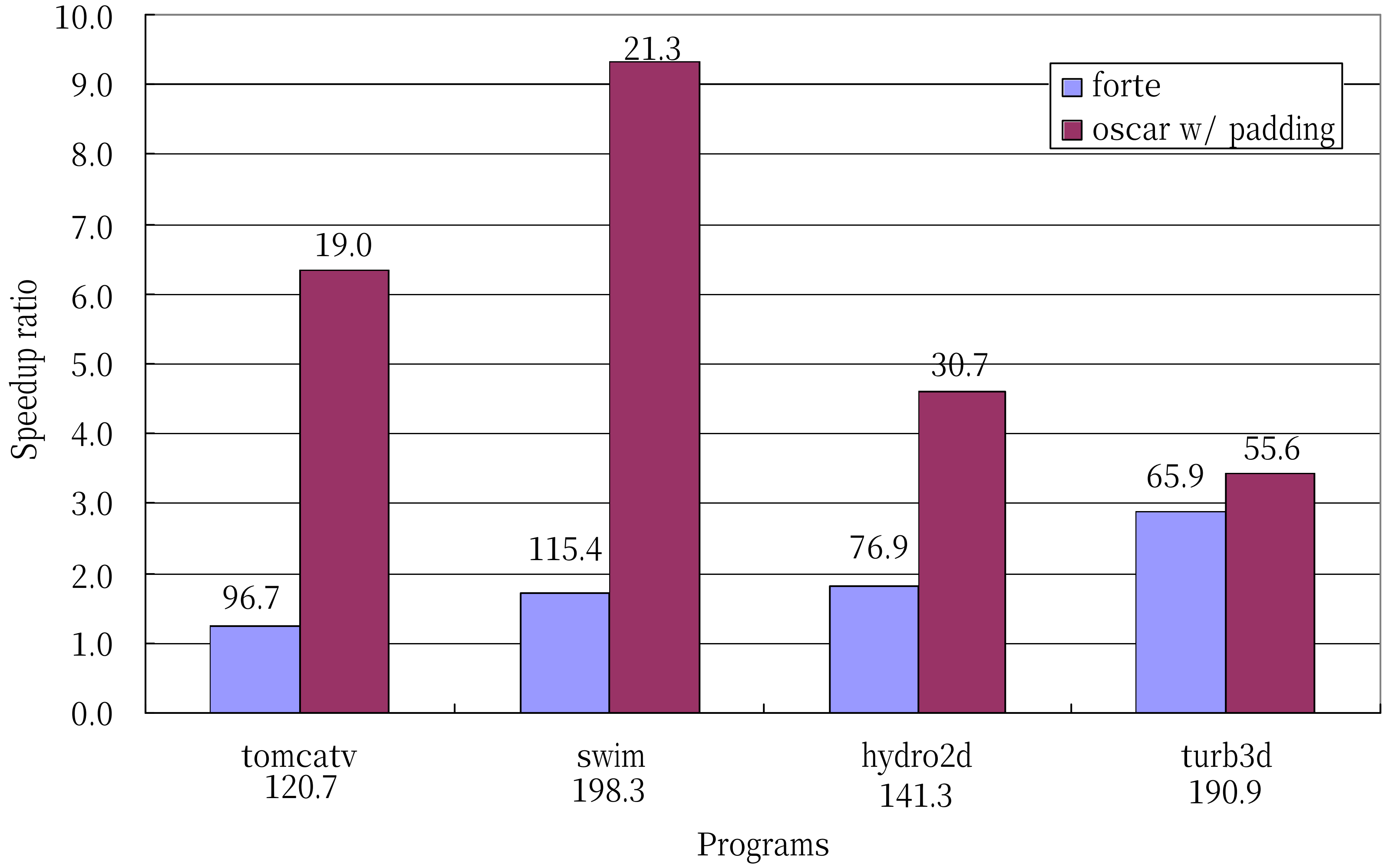
```
 PARAMETER (N1=513, N2=544)

 COMMON  U(N1,N2), V(N1,N2), P(N1,N2),
*        UNEW(N1,N2), VNEW(N1,N2),
1        PNEW(N1,N2), UOLD(N1,N2),
*        VOLD(N1,N2), POLD(N1,N2),
2        CU(N1,N2), CV(N1,N2),
*        Z(N1,N2), H(N1,N2)
```

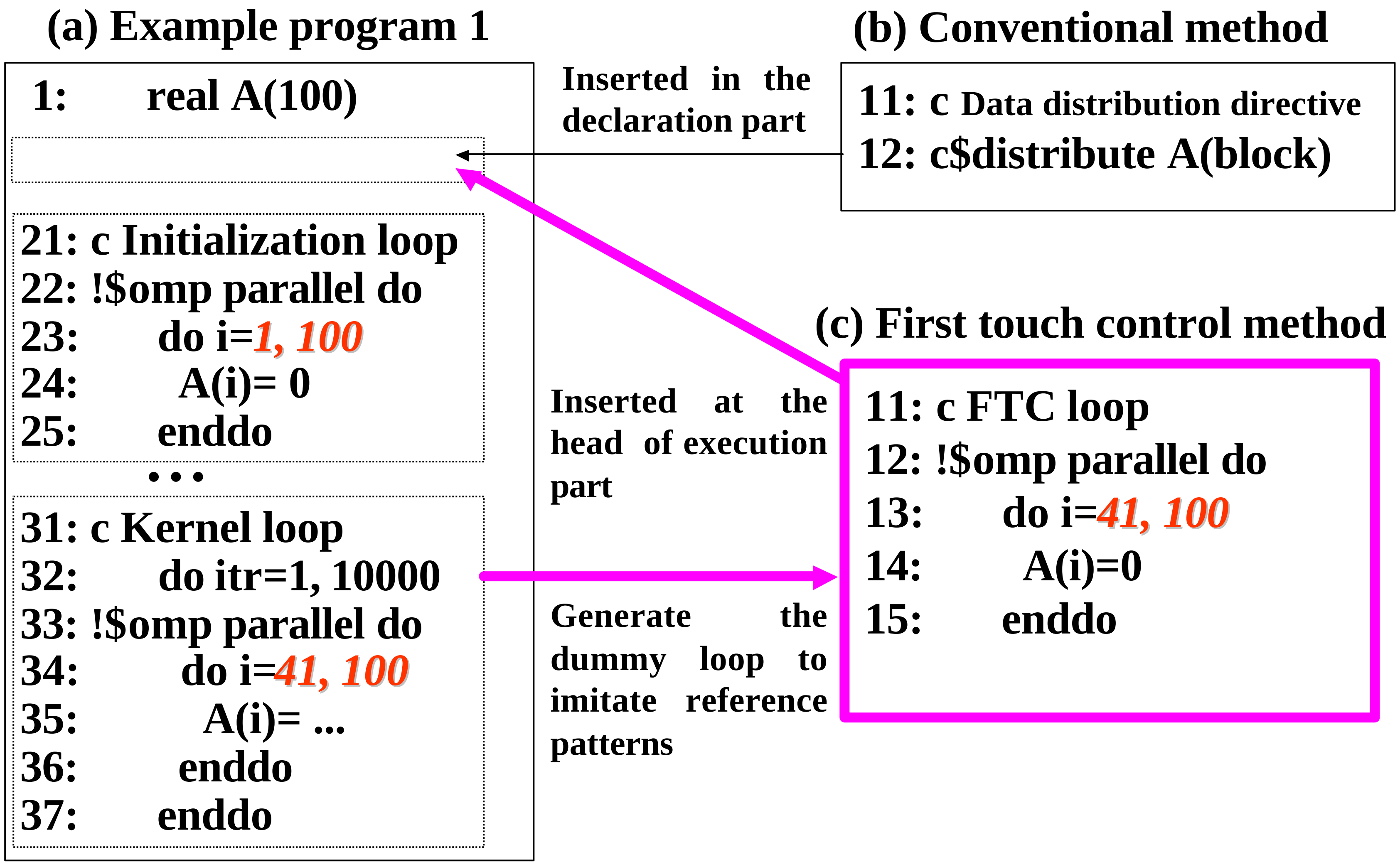


Box: Access range of DLG0

Performance of Cache Optimization with Padding on Sun Desktop WS Ultra80 (4 processors)



First Touch Control Method for Distributed Shared Memory Systems



(a) Example program 1

```
1:      real A(100)

21: c Initialization loop
22: !$omp parallel do
23:      do i=1, 100
24:        A(i)= 0
25:      enddo
            • • •
31: c Kernel loop
32:      do itr=1, 10000
33: !$omp parallel do
34:        do i=41, 100
35:          A(i)= ...
36:        enddo
37:      enddo
```

Inserted in the declaration part

(b) Conventional method

```
11: c Data distribution directive
12: c$distribute A(block)
```

Inserted at the head of execution part

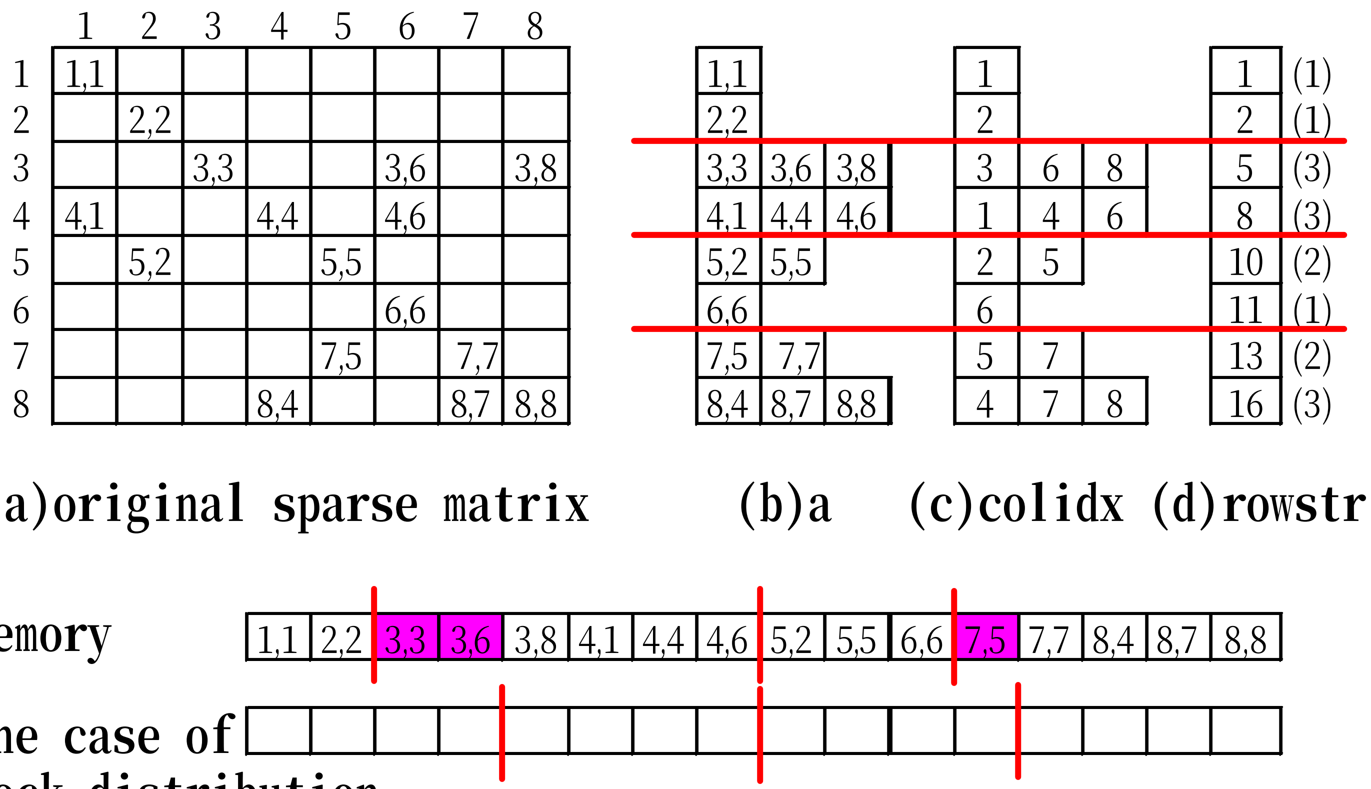
Generate the dummy loop to imitate reference patterns

(c) First touch control method

```
11: c FTC loop
12: !$omp parallel do
13:      do i=41, 100
14:        A(i)=0
15:      enddo
```

Algorithm and Data of CG

```
!$omp parallel do
do j = 1, n
  do k = rowstr(j), rowstr(j+1)-1
    sum = sum + a(k) * p(colidx(k))
  enddo
  q(j) = sum
enddo
```

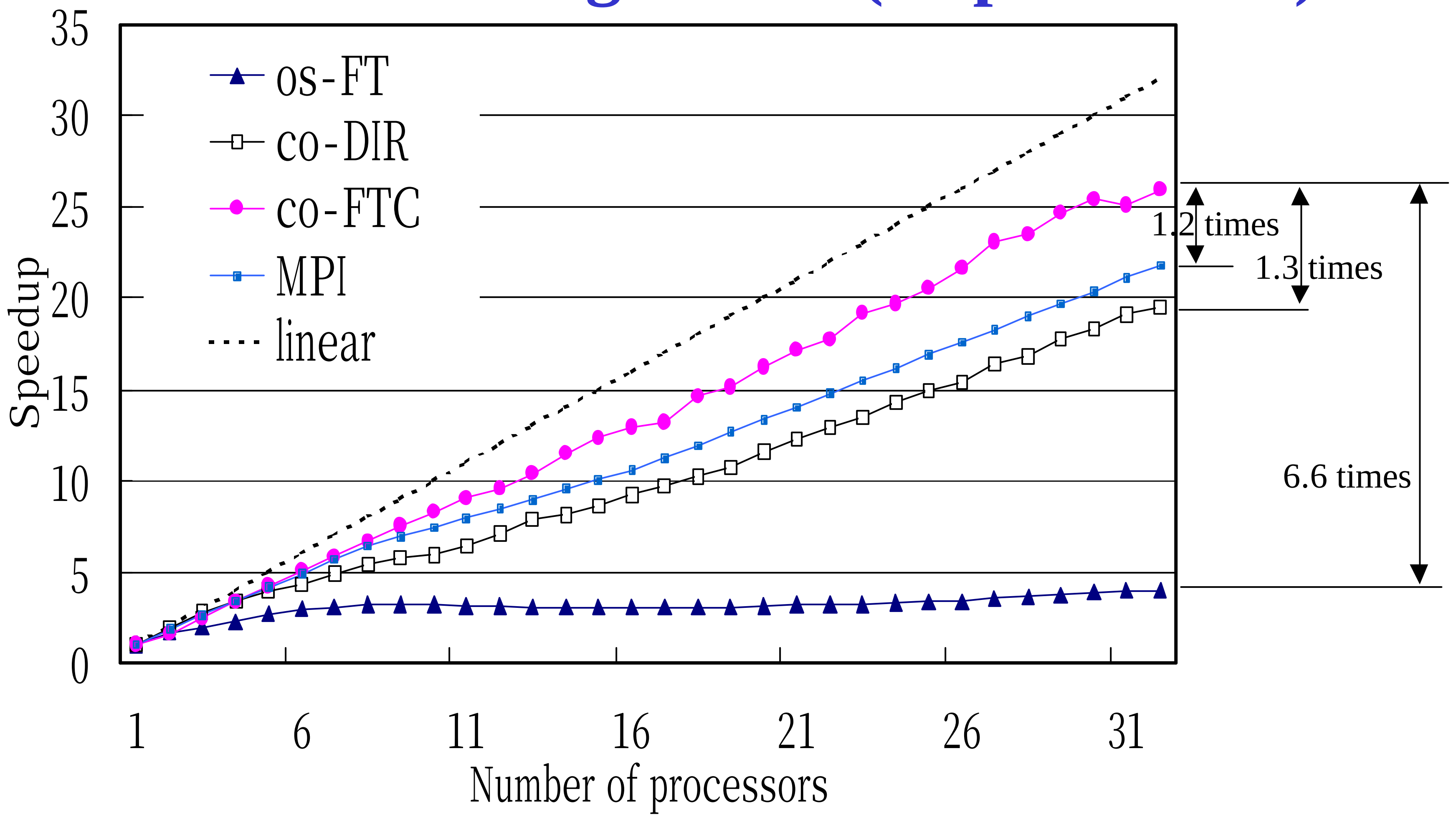


(a)original sparse matrix (b)a (c)colidx (d)rowstr

(e) on memory

(f) In the case of a block distribution

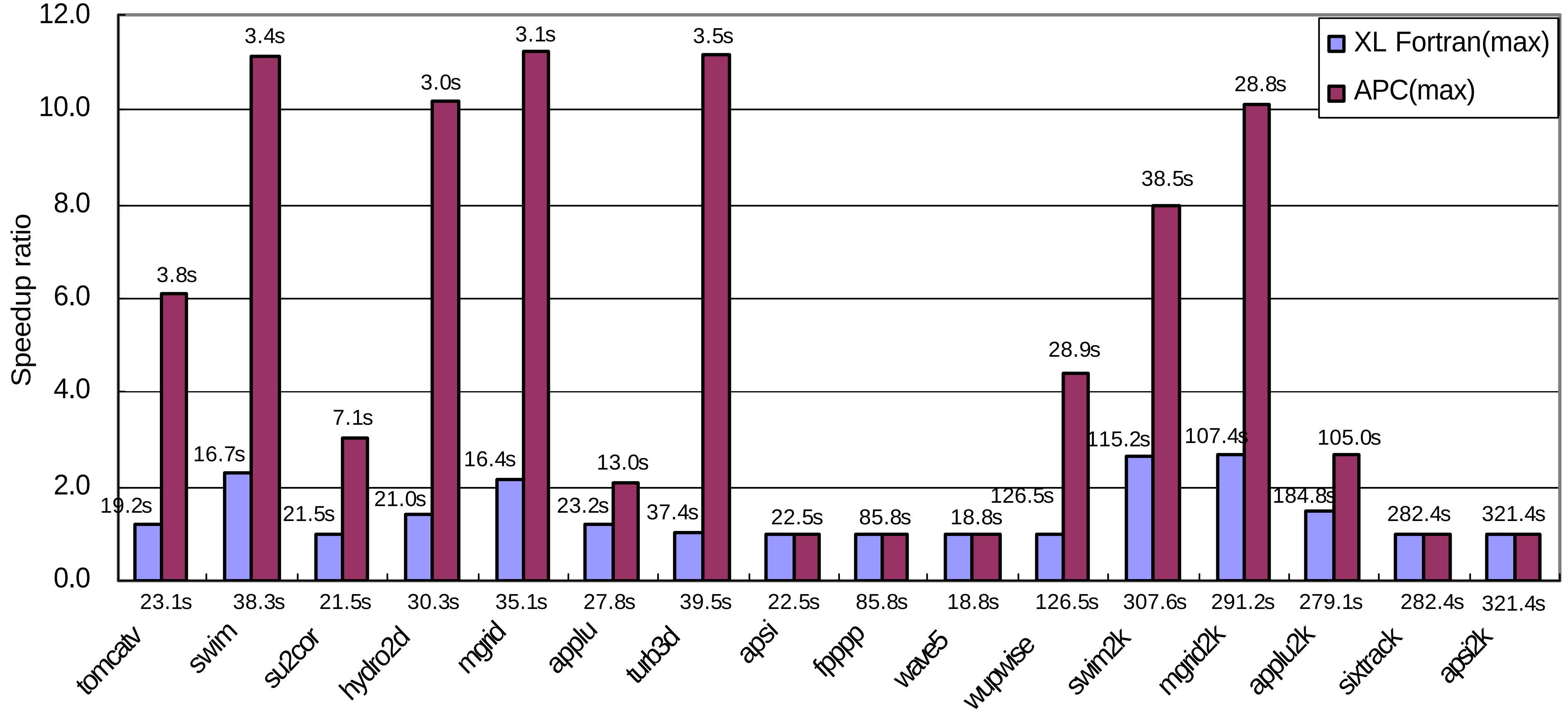
Speedup for CG (class B) by First Touch Control on Origin 2000 (32 processors)



os-FT: original, co-DIR: add data distrib. directive,
co-FTC: proposed FTC method

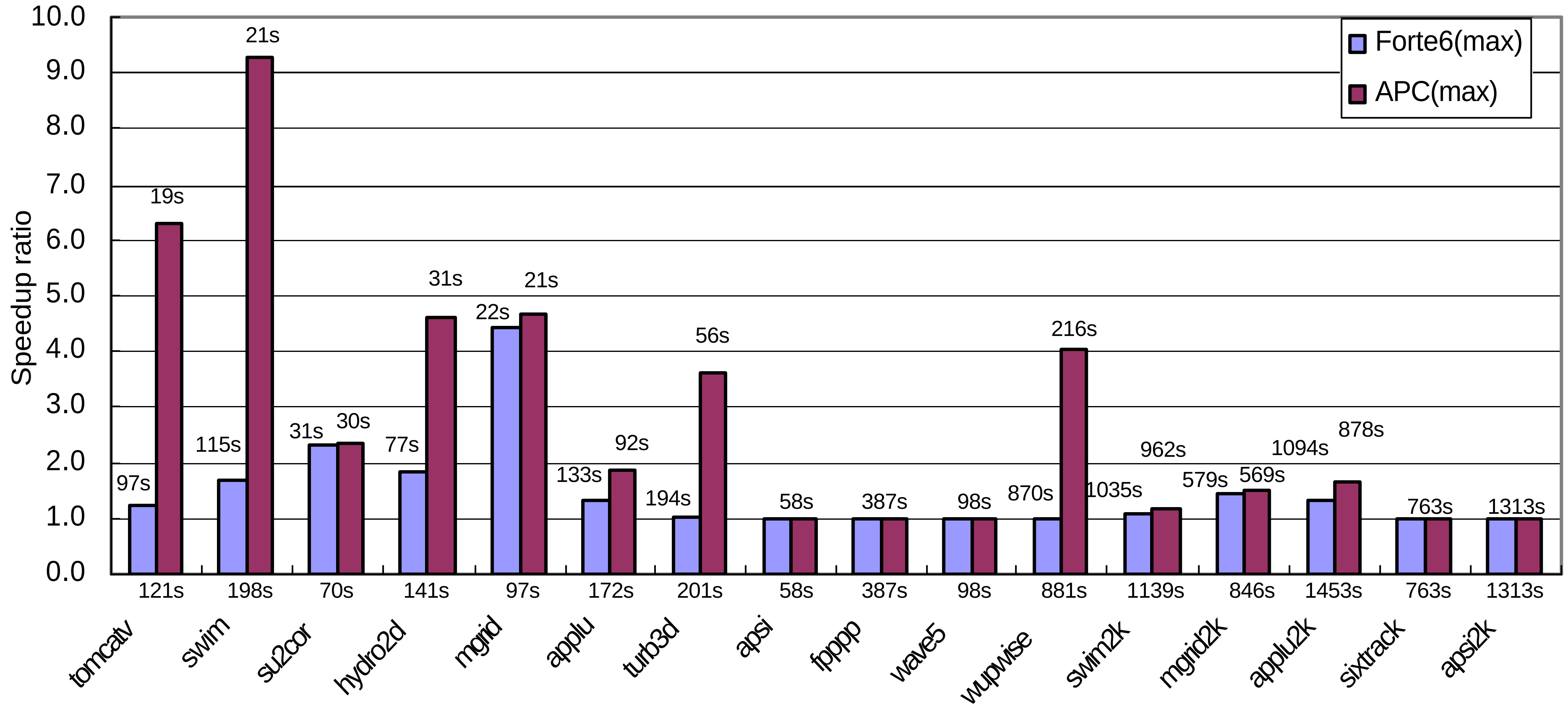
Performance of APC Compiler on IBM pSeries690 16 Processors High-end Server

- IBM XL Fortran for AIX Version 8.1
  - Sequential execution : -O5 -qarch=pwr4
  - Automatic loop parallelization : -O5 -qsmp=auto -qarch=pwr4
  - OSCAR compiler : -O5 -qsmp=noauto -qarch=pwr4 (su2cor: -O4 -qstrict)



Performance of APC Compiler on Sun Ultra80 4 Processor Workstation

- Sun Forte Developer 6 Update 2
  - Sequential execution : -fast
  - Automatic loop parallelization : -fast -autopar -reduction -stackvar
  - OSCAR compiler : -fast -explicitpar -mp=openmp -stackvar



APC Accomplishments

<Target>

Double the performance compared with product compilers for SMPs at the September 2000.

<Results & Future>

- **3.5 times speedup in average and 10.7 times at the maximum for 16 FORTRAN programs in SPEC CFP95 and CFP2000 compared with the latest compiler on the latest 16 processor high-end SMP server.**
- **45 reviewed papers, 43 technical reports,15 short papers, 8 patents, 11 invited talks, 1 invited survey paper, 10 newspaper articles, 18 Web news and 5 PhDs .**
- **Automatic multigrain parallelizing compiler will realize high productivity computing in various R&D field like Environments, Bio-informatics, Automobile, Aerospace, Financial Engineering etc.**
- **Some of the technologies will be incorporate into products.**
- **Multigrain parallelizing compiler will improve the cost performance, software and hardware productivity of chip multiprocessors to be introduced in SoC (System on Chip) like mobile phones, games, PDA, Digital TV and so on.**