Cache Optimization using Data Localization for Multigrain Parallel Processing

K. Ishizaka[12], M. Obata[12], K. Kimura[12], H. Nakano[12], A. Yoshida[13], H. Kasahara[12]

([1] APC Technology Group, [2] Waseda University, [3] Toho University)

Research Background

- Wide use of SMP machine
  - From chip multiprocessor to high performance computer
- Increasing the number of processors
  - Gap between peak and effective performance is getting larger
- To increase effective performance further
  - Multigrain parallel processing
    - Beyond limitation of loop parallelization
  - Effective use of cache
    - Cope with the speed gap between processor and memory

Related Work

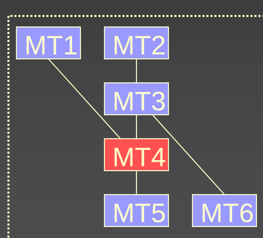
- SUIF Compiler
  - Data and Computation Transformations
    - Anderson et al. PPoPP95
  - Compiler-directed page coloring
    - Bugnion et al. ASPLOS96
- Decreasing cache miss by Padding
  - Cache miss equation
    - Ghosh et al. ICS97
  - Genetic algorithm for cache miss equation
    - Vera et al. LCPC2002
  - Padding heuristics
    - Rivera et al. SC99

Data Localization

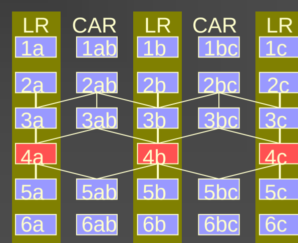
- Exploit data locality to use faster memory effectively
- Execution order transformation
  - Loops sharing large data are decomposed into smaller loops by Loop Aligned Decomposition(LAD) considering cache size
  - Change execution order of decomposed loops to exploit data locality considering Earliest Executable Condition by static or dynamic scheduling
- Data layout transformation
  - Change data layout to decrease cache line conflicts by padding

Loop Aligned Decomposition

- Detection of Target-Loop-Group (TLG) from MTG
- Loops inside TLG are decomposed into LR and CAR
  - LR (Localizable Region)
    - where data are transferred via cache or LM (DSM)
  - CAR (Commonly Accessed Region)
    - a set of iterations on which different LRs are commonly data-dependent



Example of TLG

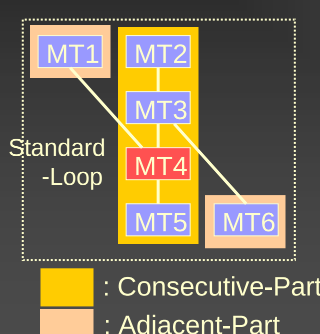


Note: Data-dep. to MT1 and MT6 are omitted

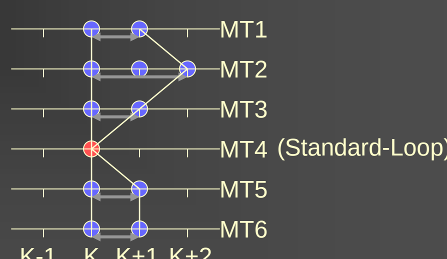
MTG after LAD

Inter Loop Dependence Analysis in TLG

- Inter Loop Dependence (ILD)
  - Inter-Iteration data dependence over different loops in TLG



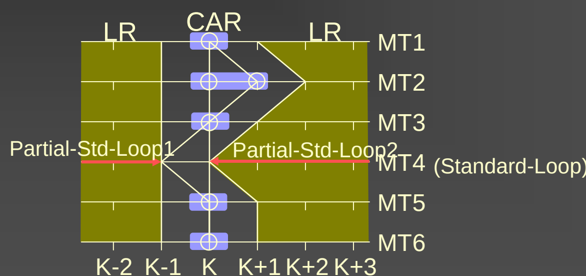
Example of TLG



Result of ILD analysis

Decomposition of TLG into CAR and LR

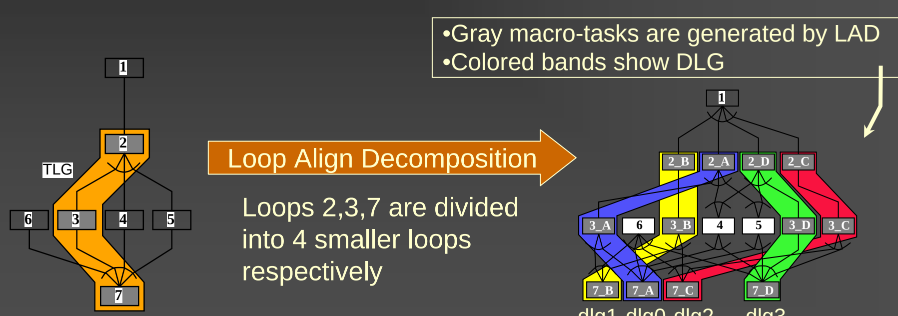
- Decompose Standard-loop into N Partial-Standard-Loops
  - N is determined considering cache size.
- Define iterations on which Partial-Standard-Loops are commonly data dependent as CAR
- Define iterations except CAR as LR



○: Iteration of CAR

Data Localizable Group (DLG)

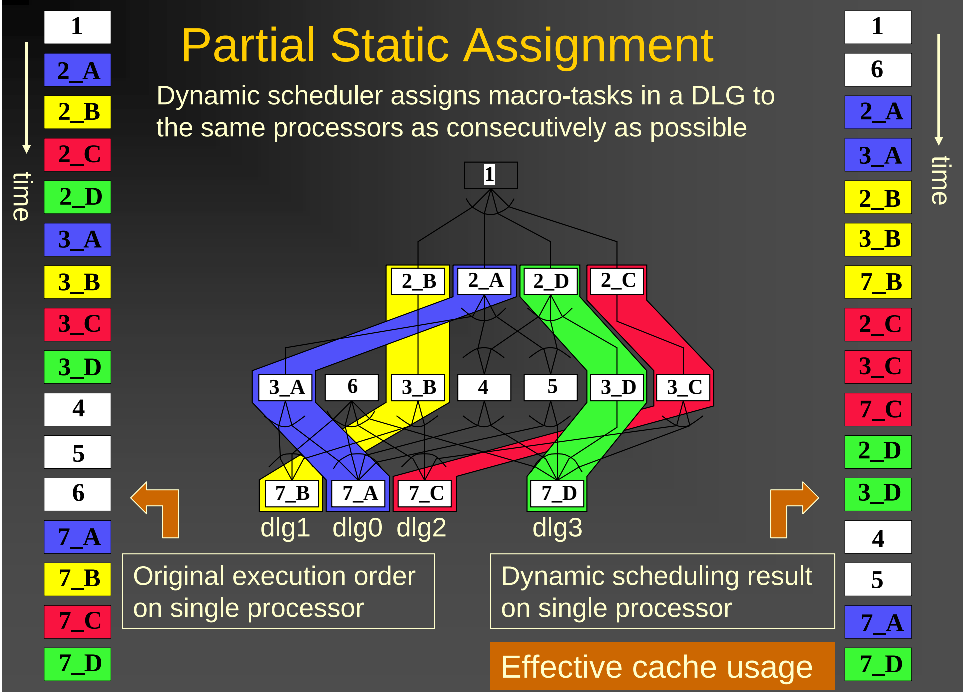
- Define Data Localizable Group (DLG)
  - DLG: group of macro-tasks to be assigned to the same thread for passing the shared data through cache



(a) Before loop decomposition (b) after loop decomposition

Partial Static Assignment

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



Data Layout Transformation

- Data size accessed by loops in one DLG is smaller than cache size
- Loops in DLG are executed consecutively
- Padding for DLG
  - remove line conflicts among data in DLG by inter-array padding
- Caches are used effectively over loops in DLG

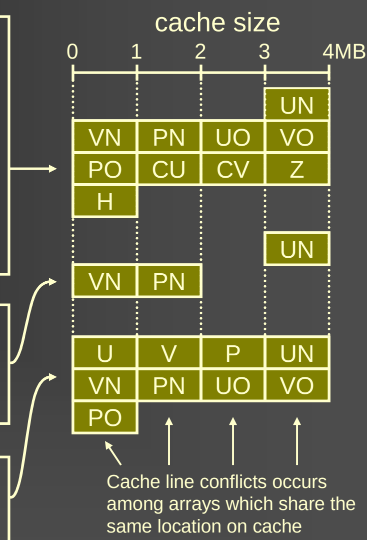
An Example of Data Localization for Spec95 Swim

```
      DO 200 J=1,N
      DO 200 I=1,M
         UNEW(I+1,J) = UOLD(I+1,J)+
     1   TDTS8*(Z(I+1,J+1)+Z(I+1,J))*(CV(I+1,J+1)+CV(I,J+1)+CV(I,J)
     2      +CV(I+1,J))-TDTSDX*(H(I+1,J)-H(I,J))
         VNEW(I,J+1) = VOLD(I,J+1)-TDTS8*(Z(I+1,J+1)+Z(I,J+1))
     1      *(CU(I+1,J+1)+CU(I,J+1)+CU(I,J)+CU(I+1,J))
     2      -TDTSDY*(H(I,J+1)-H(I,J))
         PNEW(I,J) = POLD(I,J)-TDTSDX*(CU(I+1,J)-CU(I,J))
     1      -TDTSDY*(CV(I,J+1)-CV(I,J))
  200 CONTINUE
```

```
      DO 210 J=1,N
         UNEW(1,J) = UNEW(M+1,J)
         VNEW(M+1,J+1) = VNEW(1,J+1)
         PNEW(M+1,J) = PNEW(1,J)
  210 CONTINUE
```

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

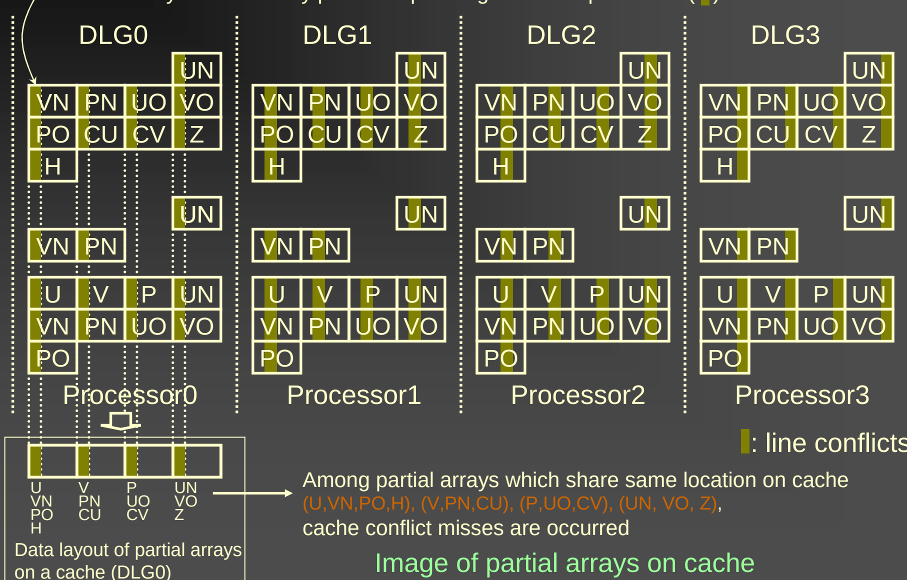
(a) An example of target loop group for data localization



(b) Image of alignment of arrays on cache accessed by target loops

Line Conflicts in Data Localization on 4 Processors

- Loops are decomposed into 4 partial loops
- Partial arrays accessed by partial loops assigned same processor ( )

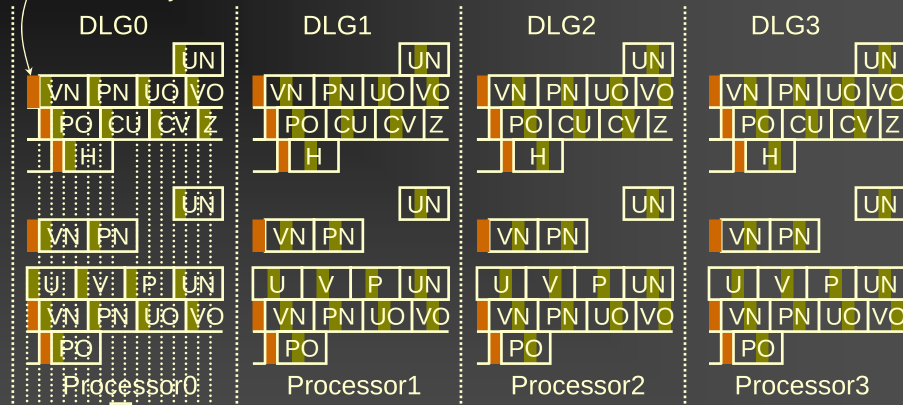


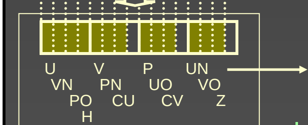
Among partial arrays which share same location on cache (U,VN,PO,H), (V,PN,CU), (P,UO,CV), (UN, VO, Z), cache conflict misses are occurred

Image of partial arrays on cache

Remove Line Conflicts in DLG by Padding

- Padding: Inter-array padding to remove conflict among partial arrays accessed by DLG





Data layout of partial arrays on a cache (DLG0)

- Padding removes conflicts
- Cache is used more effectively

: padding

Image of partial arrays on cache after padding for DLG

Data Layout for Removing Line Conflict Misses by Array Dimension Padding

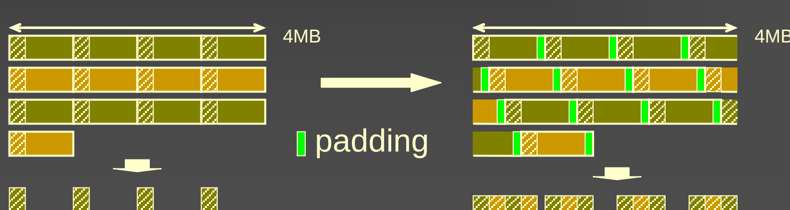
Declaration part of arrays in spec95 swim

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

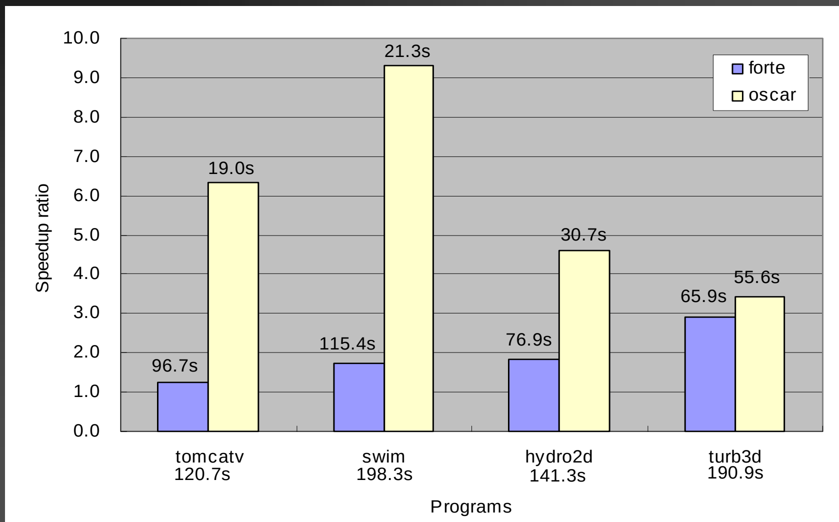
```
PARAMETER (N1=513, N2=544)

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)

450MHz Ultra SPARC II, L1:16KB/L2:4MB, Solaris 8, Forte 6 update 2



Conclusions

- Cache Optimization using Data localization
  - Improve data locality over loops
  - Minimize cache misses by inter array padding

- 5.1, 5.5, 2.5, 1.2 times speedup against Forte 6 update 2 compiler for SPEC CFP95 tomcatv, swim, hydro2d and turb3d on 4processors workstation Sun Ultra80