

Parallel Reduction: An Application of Adaptive Algorithm Selection

Hao Yu, Francis Dang and Lawrence Rauchwerger

Department of Computer Science,
Texas A&M University
<http://www.cs.tamu.edu/people/rwenger,fhd4244,h0y8494>
h0y8494,fhd4244,rwenger@cs.tamu.edu

Motivation

Question:

Which algorithm is the most appropriate for a given (application-input) tuple ?

Solution:

1. Extract attributes characterizing applications and inputs.
2. Bridge (Map) the attributes and the algorithms.
3. For a given application and input, predict the most appropriate optimization algorithm.
4. Apply the chosen algorithm, at run-time.

Contributions [ICS00]:

- A systematic process of algorithm characterization.
- Ability to predict and select the best available algorithms for reduction parallelization.

Reduction and Irregular Reduction

Reduction is

- Associative and commutative operation: $x = x \otimes \text{exp}$
- x does not occur in the exp or anywhere else in the loop (except for other reduction statements.)
- x can be scalar, array reference, or vector (in FORTRAN 90).

Irregular Reduction

- Refer to the reductions in the following example

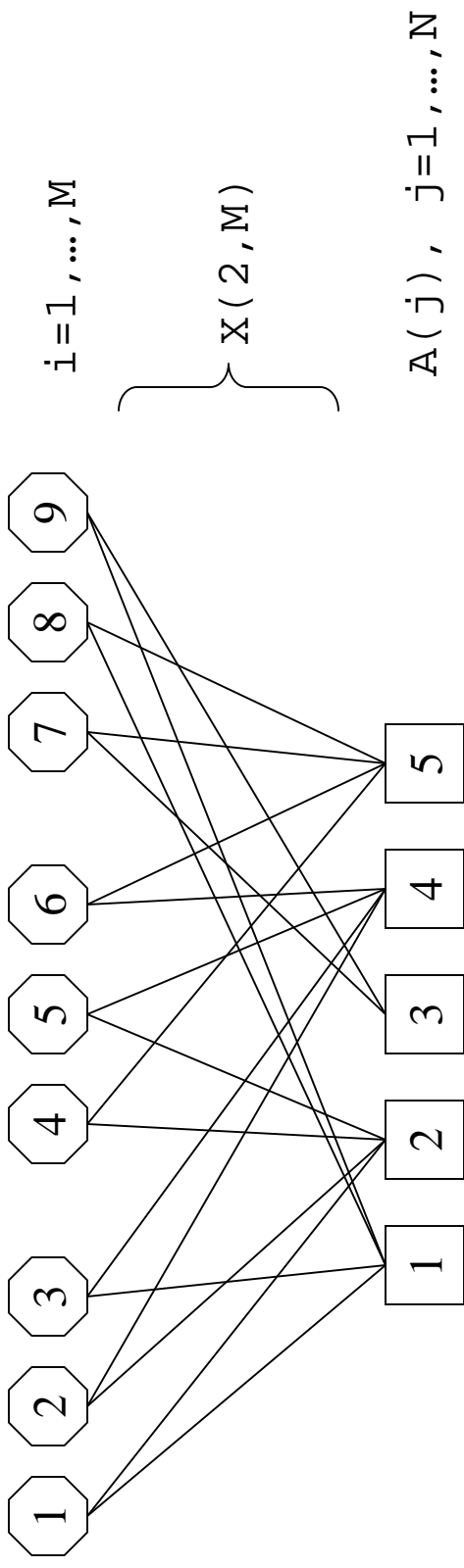
```
integer X(1:M)      do I = 1, M
real   A(1:N)      A(X( I )) = A(X( I )) ⊗ exp
                                     enddo
```

- Access pattern can not be extracted at compile-time.

Reduction Parallelization

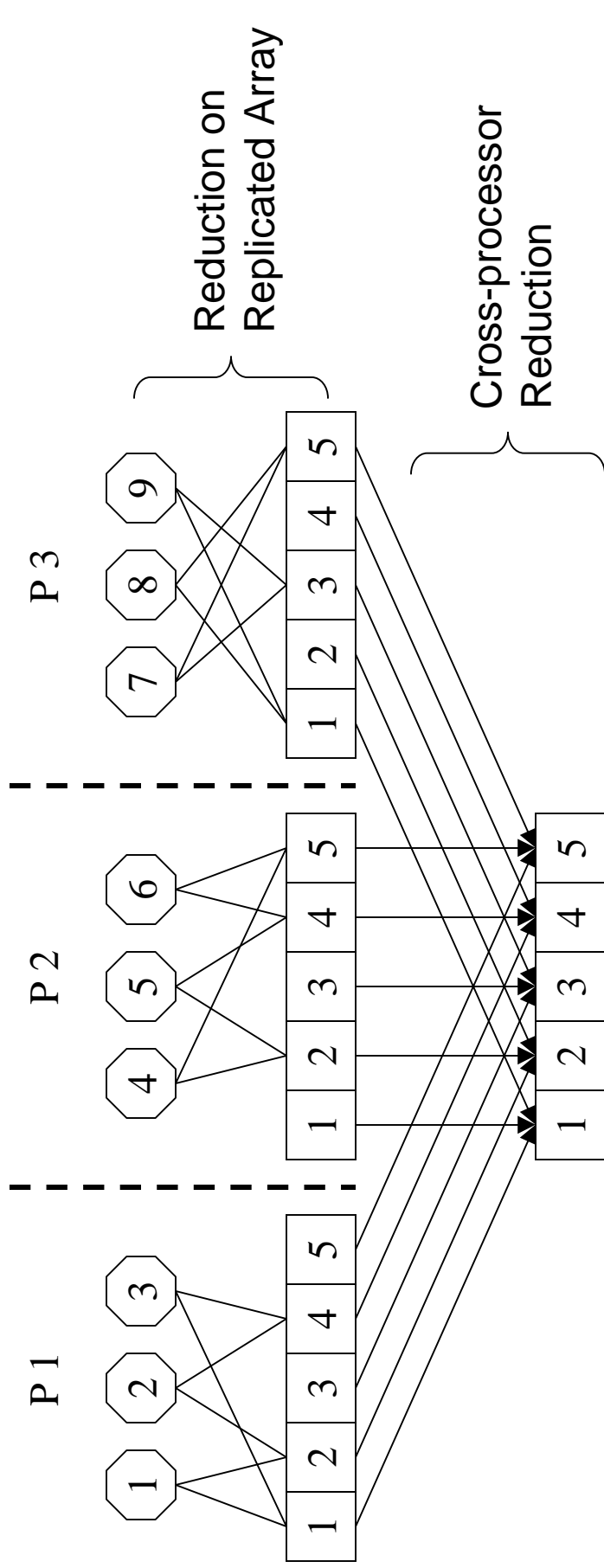
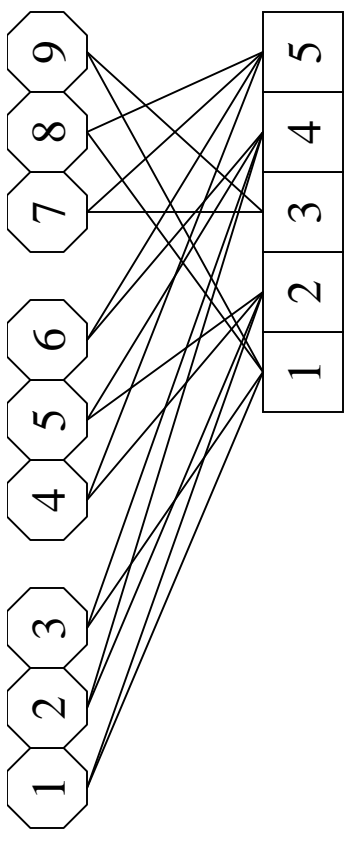
```
do i = 1, M
  A(X( 1,i )) = A(X( 1,i )) + exp1( )
  A(X( 2,i )) = A(X( 2,i )) + exp2( )
enddo
```

Serial Reduction



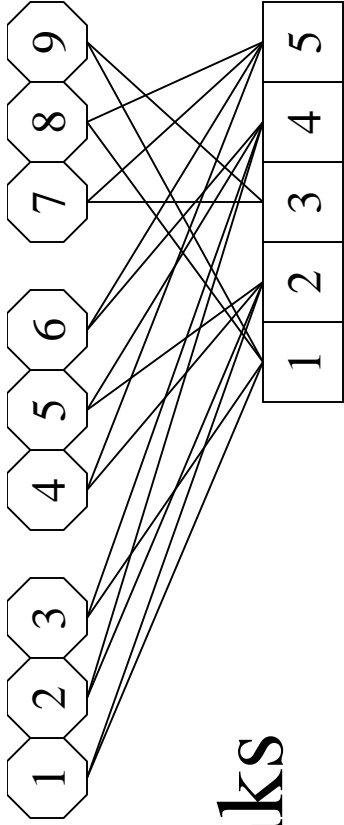
[Han and Tseng, IPDPS01]

Replicated Buffer

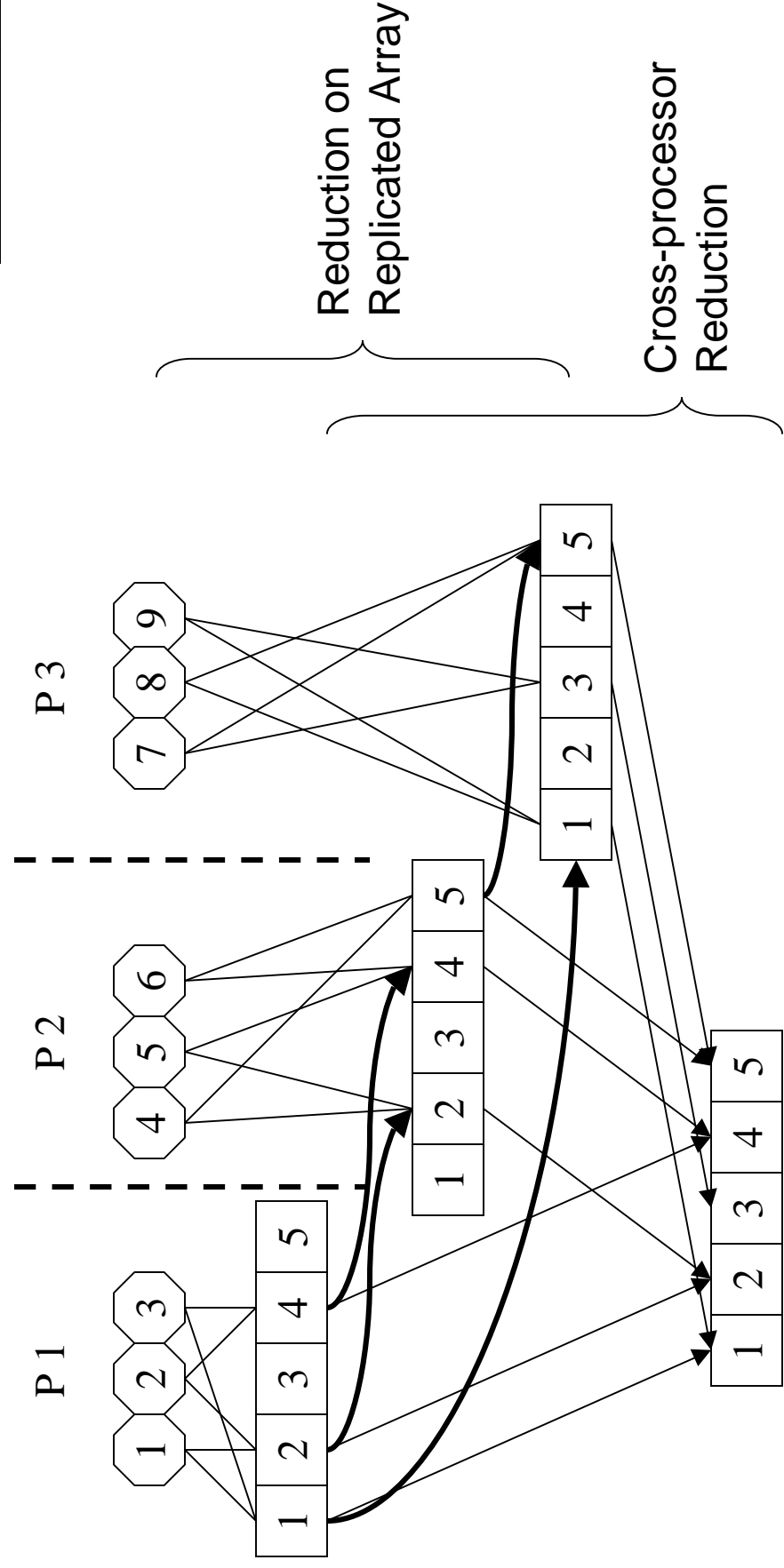


Pros: Simple

Cons: Not scalable for sparse access patterns

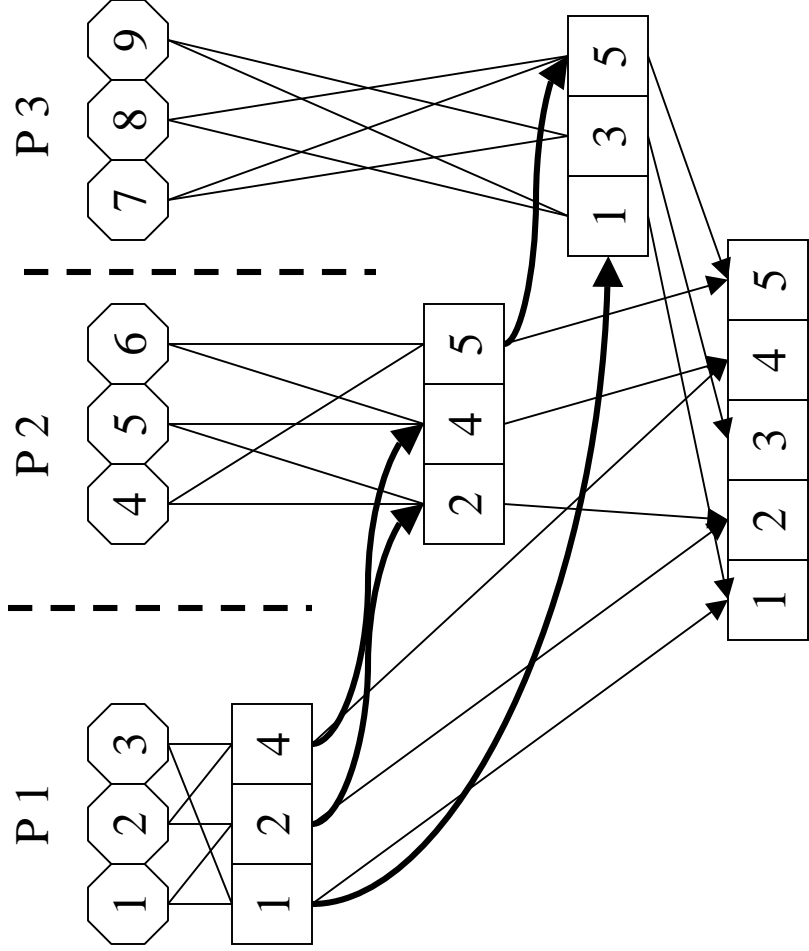
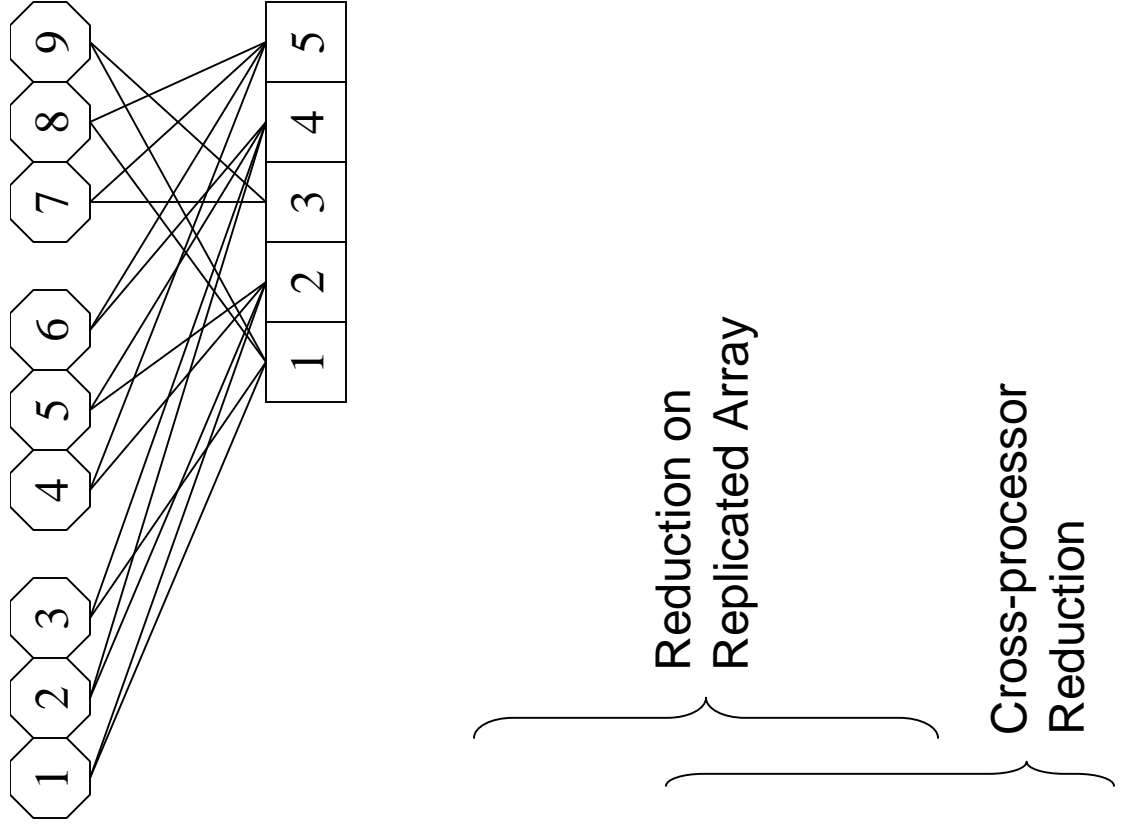


Replicated Buffer with Links



- Pros:* Reduce unnecessary references on Replicated array.
- Cons:* Extra memory to keep links.

Selective Privatization

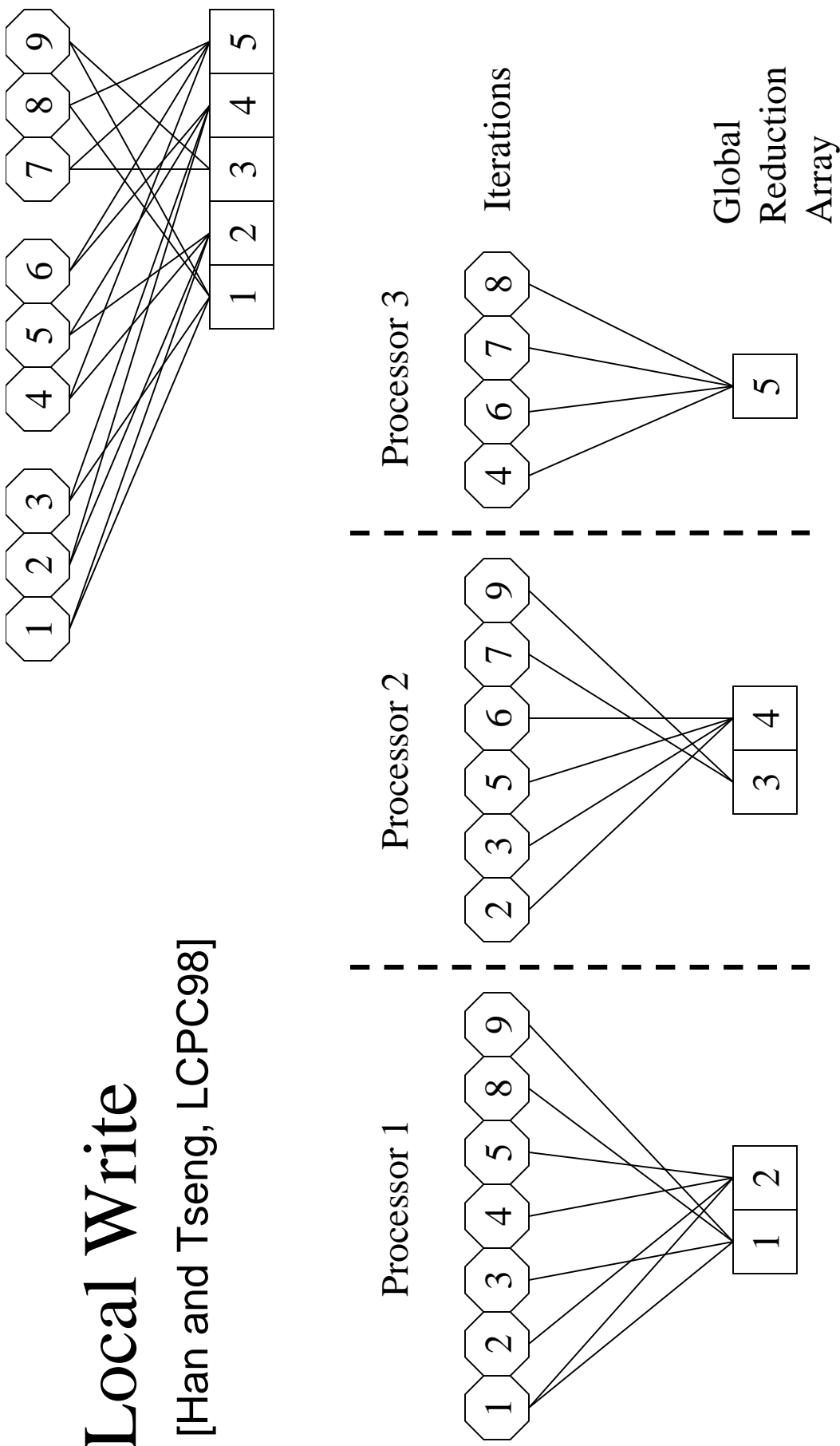


Pros: dense private space

Cons: Setup phase is proportional to the number of iterations.

Local Write

[Han and Tseng, LCPC98]



Pros: Increases locality (following 'owner computes' rule)

Cons: Needs inspector loop; may heavily replicate work

Comparison of Schemes

Issues	RepBufs	RepLink	SelfPriv	LocalWrite
Sensitive to pattern change?	NO	YES	YES	YES
Inspector complexity		$O(N)$	$O(M)$	$O(M)$
Preferred pattern	Dense	Sparse	Sparse	No preference
Locality	Poor	Poor	Good	The best
Extra Space (worst case)	$O(N*P)$	$O(N*P)$	$O(N*P+M)$	$O(M*P)$
Replicated Work?	NO	NO	NO	YES

M: Number of Iterations
 N: Size of Reduction Array
 P: Number of Processors

- No single parallelization scheme fits all access patterns
- Attributes can be extracted for modeling and ranking different schemes.

Attributes

```
REAL A(N), pA(N,P)
INTEGER X(2,M)

DOALL i = 1, M
  U1 = func1()
  U2 = func2()
  pA(X(1, i), ip) = pA(X(1, i), ip) + U1
  pA(X(2, i), ip) = pA(X(2, i), ip) + U2

DOALL i = 1, N
  A(i) = A(i) + pA(I, 1:P )
```

N

Dimension of Reduction Array

CONnectivity

Number of Iterations (M)

Number of distinct memory references (N)

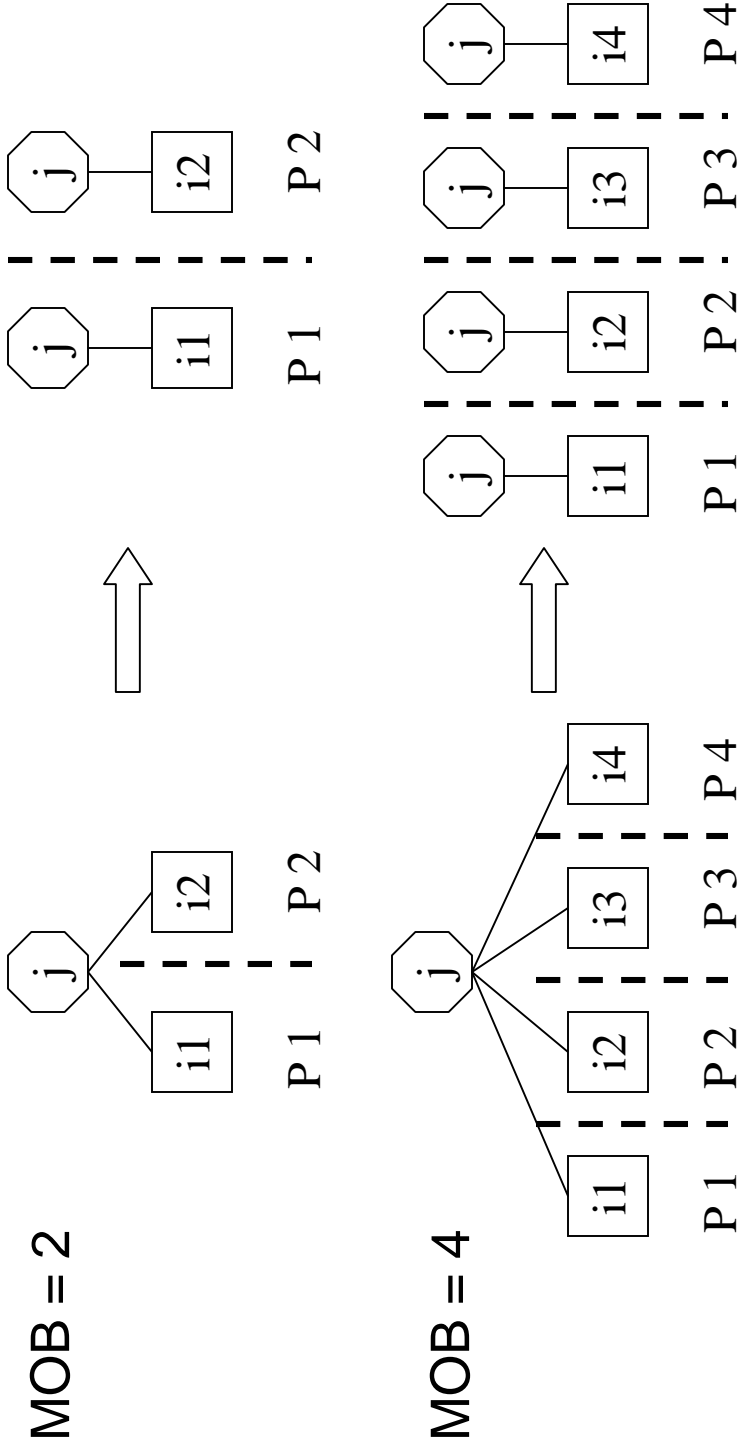
OTHer work

$$\frac{T_{loop} - T_{reductions}}{T_{reductions}}$$

Attributes

MOBility *Number of distinct reduction elements
(referenced in one iteration)*

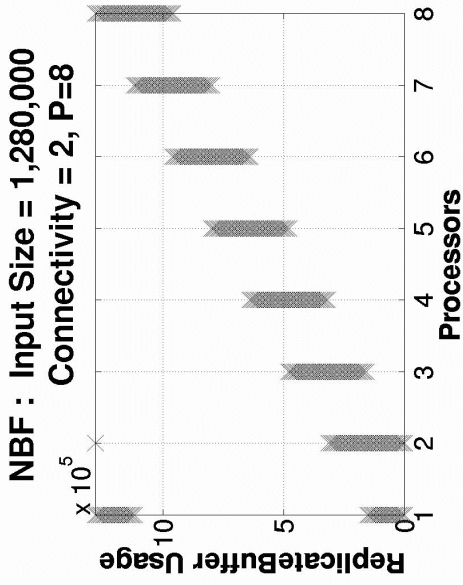
MOB affects iteration replication rate of Local Write



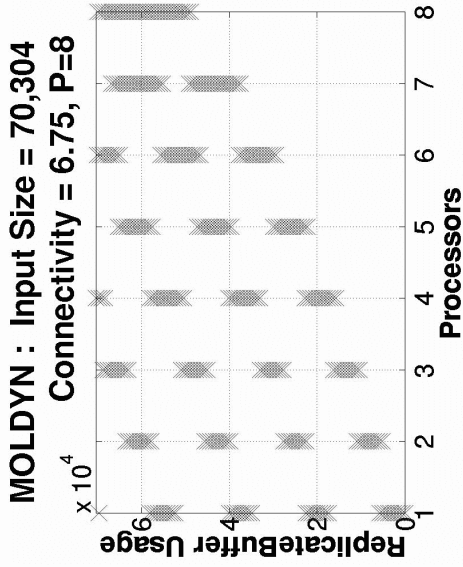
Attributes

$$\text{CHR} = \frac{\text{Number of touched elements of replicated array}}{\text{Size of replicated array}}$$

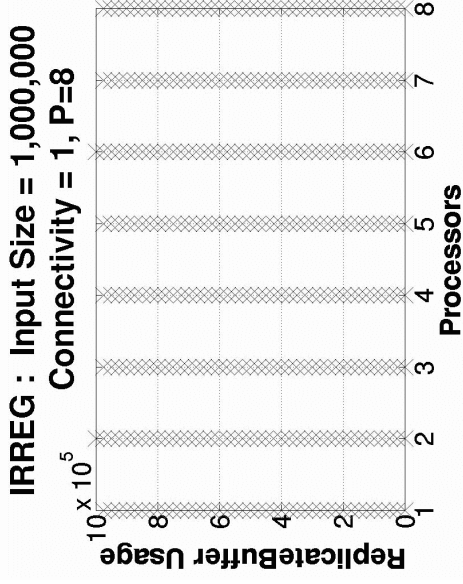
CLUSsterness Degree of Clustering to classify whether the touched private elements are scattered or clustered on every processor.



CHR	0.25
CLUS	1



CHR	0.33
CLUS	2



CHR	0.26
CLUS	3

Decoupled Effects of Attributes

Attributes	RepBufs	SelPriv	LocalWrite
N		↑	↑
CONnectivity	↑	↑	
OTHer work	↑	↑	▲
MOBility			▲
CHR	▲	▲	▲
CLUSterness		▲	

 Positive Effect
  Little Positive Effect
  Negative Effect

- Monotonic effect on performance.
- Different effect on different schemes.

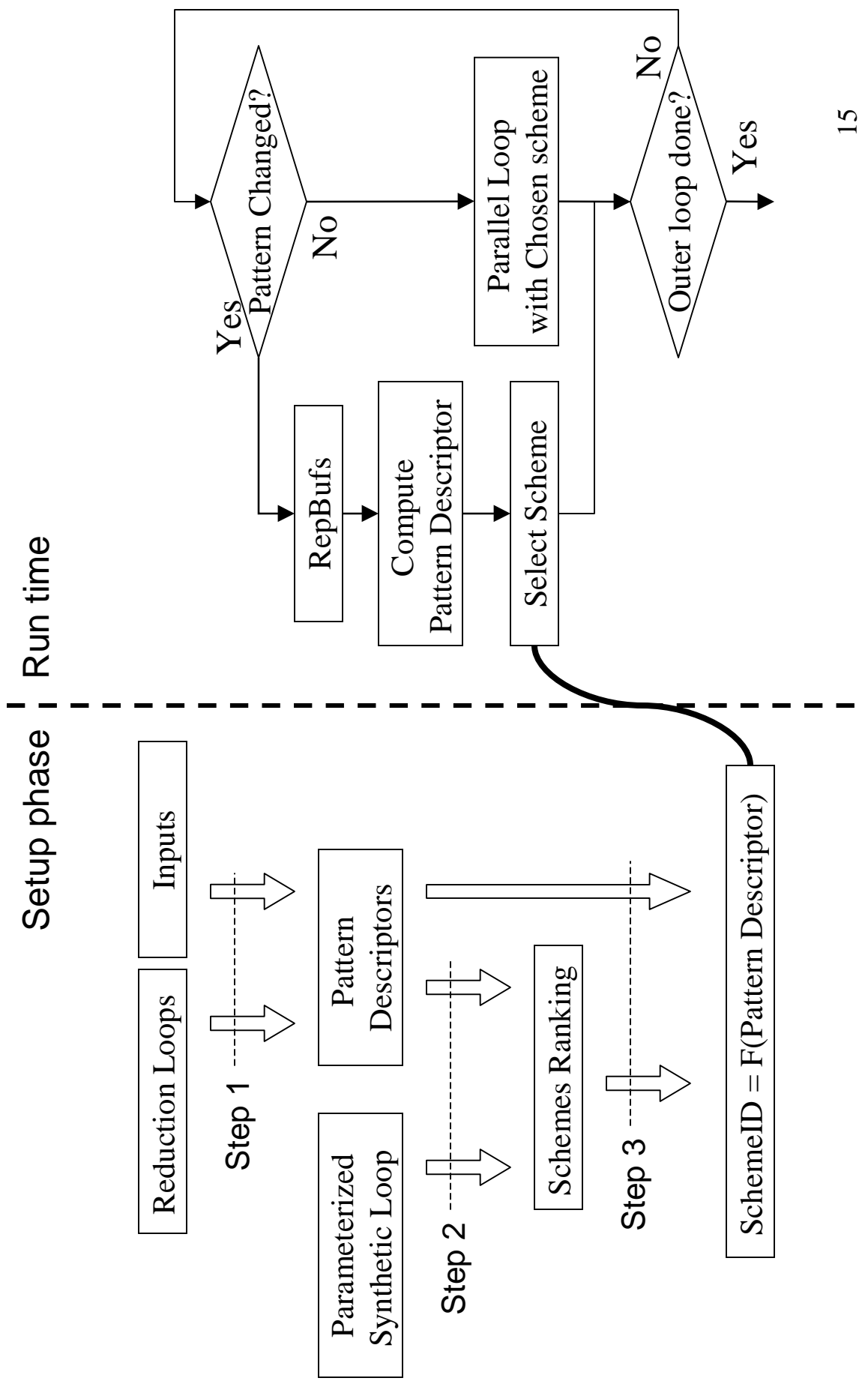
Pattern Descriptor

Pattern Descriptor

- Vector of these Attributes
- The attributes describes various characteristics of a reduction loop.
- The characteristics matter most to model and rank reduction parallelization schemes in term of performance.

Adaptive Algorithm Selection

Overall Process



Setup: Factorial Experiment

A List of Values of Pattern Descriptor

- Select several values for each attribute.
- Generate pattern descriptors from all combinations of these values.

Synthetic Loop

```
DO j = 1, N*CON
  DO i = 1, OTH
    ! Some computation
  ENDDO
  DO i = 1, MOB
    k = index(i, j)
    data(k) += foo()
  ENDDO
ENDDO
```

Sample Generation

```
FOR (each pattern descriptor) DO
  Generate index array
  FOR (each scheme) DO
    Instrument synthetic loop
    Execute parallel loop
    Measure speedup
  ENDFOR
  Rank schemes
ENDFOR
```

Model Generation 1: Statistical Regression

Setup Phase

Input: Samples

Output: $F[1:K]$, K = number of schemes

```
FOR (each scheme S) DO
  FOR (each candidate polynomials  $F_i$ ) DO
    fit for Speedup(S) =  $F_i$ (Pattern Descriptor)
  F[S] =  $F_i$  with minimal error
```

Run-time Scheme Selection

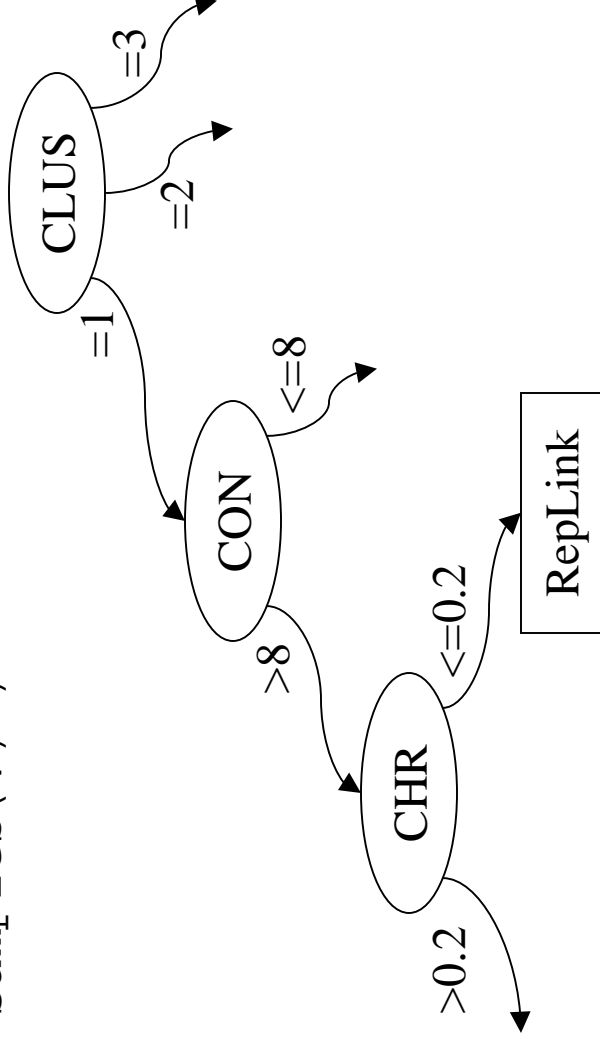
```
FOR (each scheme S) DO
  Speedup(S) =  $F[S]$ (Pattern Descriptor)
Scheme = S with maximal Speedup(S)
```

Model Generation 2: Decision Tree Learning

Algorithm (BuildTree)

```
Input: Samples; Output:Tree
Attr = The best attribute to partition Samples
Tree = node to test Attr.
FOR (each value(interval) V of Attr) DO
  IF ( STOP( Samples(V) ) THEN
    Build LEAF node
  ELSE
    BuildTree( Samples(V) )
```

An Example



Experimental Results - Setup

- Experimental environment:
 - 16 processor HP V-class
 - 4GB memory
 - HPUX11 operating system.
- Implemented Reduction Algorithms:

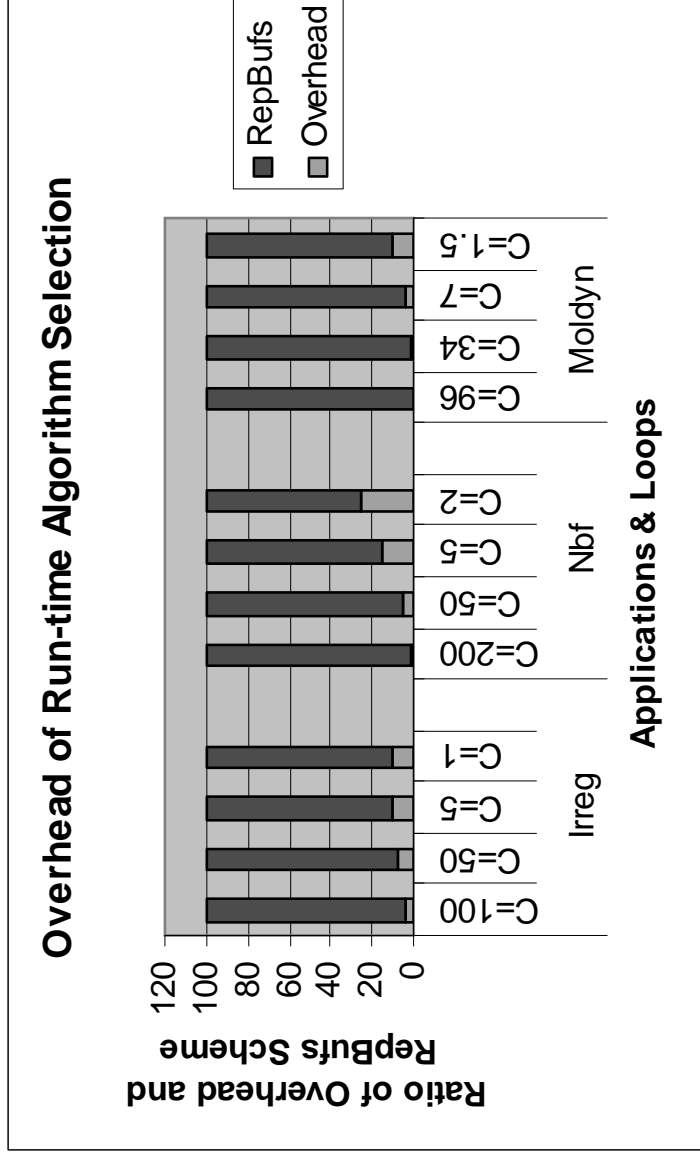
Replicated Buffer
Replicated Buffer with Links
Selective Privatization
Local Write

- Used Software:
 - SAS/STAT
 - C4.5
- Codes and Loops:

Application	Loop
Irreg	DO 100
NBF	DO 50
Moldyn	ComputeForces loop
Charmm	DO 78
Spark98	smvpthread() loop
SPICE 2G6	BJT (GOTO 100) loop
Fma3d	Quadrilateral Plate loop in Scatter_Element_Nodal_Forces

Experimental Results

Run-time Overhead



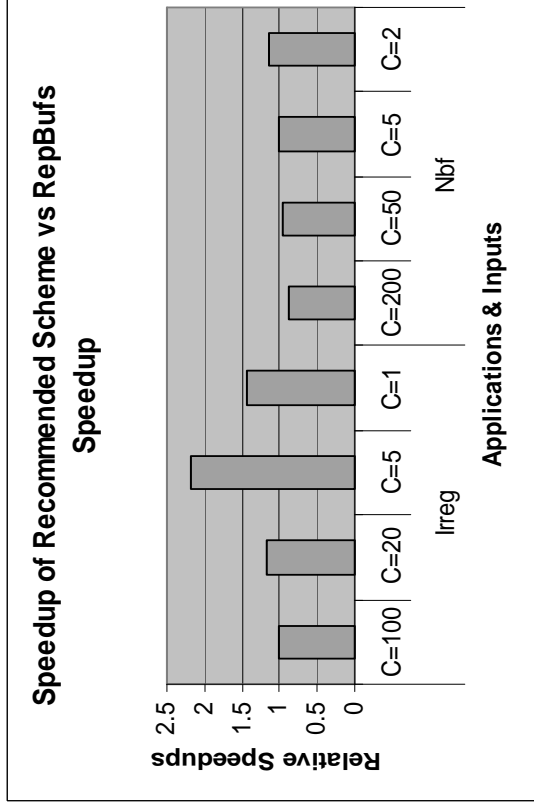
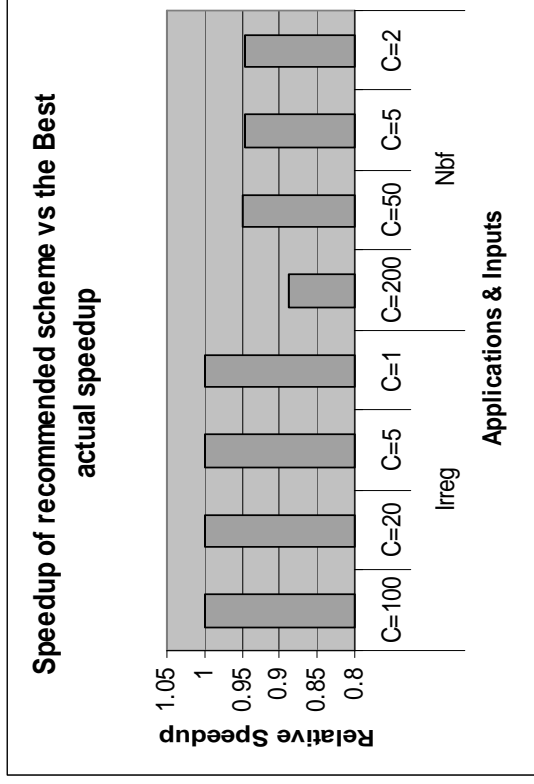
Run-time Overhead Includes:

1. Computing run-time attributes of pattern descriptor: N, CON, CLUS, CHR
2. Selecting the best scheme either by evaluating a polynomial or by visiting the decision tree with maximal height as 6.

Experimental Results

Irreg & Nbf

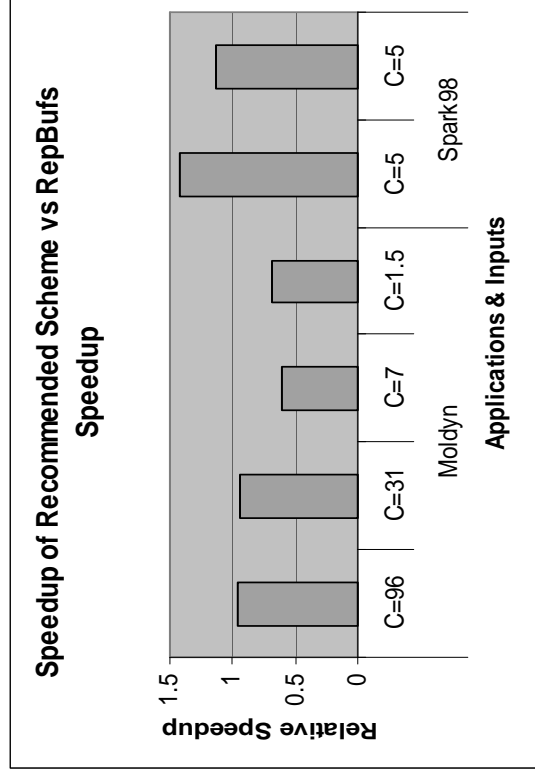
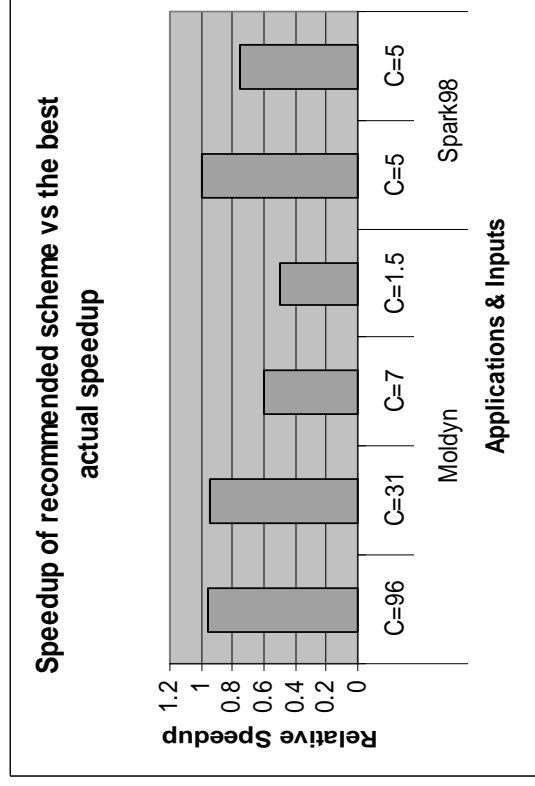
App	N	CON	MOB	OTH	CHR	CLUS	Recommended Scheme		Experimental Result
							Regression	Decision Tree	
Irreg	100,000	100	2	0	0.92	1	REP	SEL	REP
	500,000	20			0.71	1	SEL	SEL	SEL
	1,000,000	5			0.40	3	LOW	LOW	LOW
	2,000,000	1			0.26	3	SEL	LOW	SEL
Nbf	25,600	200	2	0	0.25	1	RLL	SEL	REP
	128,000	50			0.25	1	RLL	SEL	REP
	256,000	5			0.25	1	SEL	SEL	RLL
	1,280,00	2			0.25	1	SEL	SEL	RLL



Experimental Results

Moldyn & Spark98

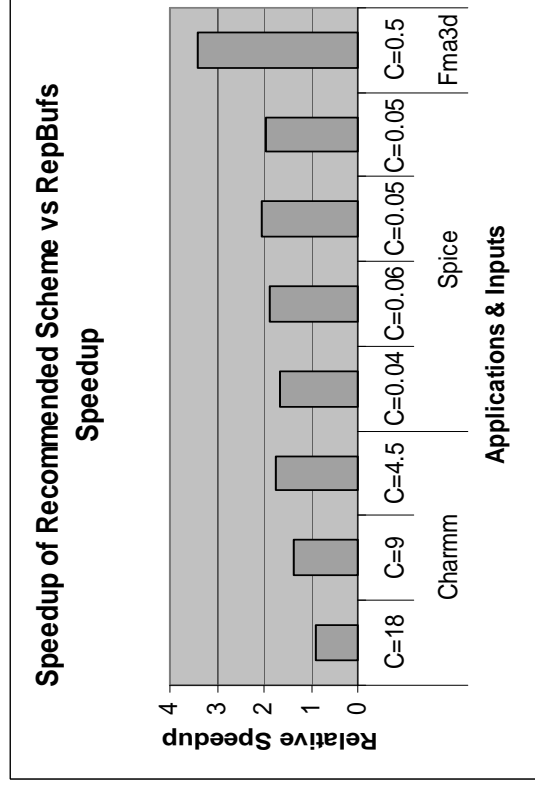
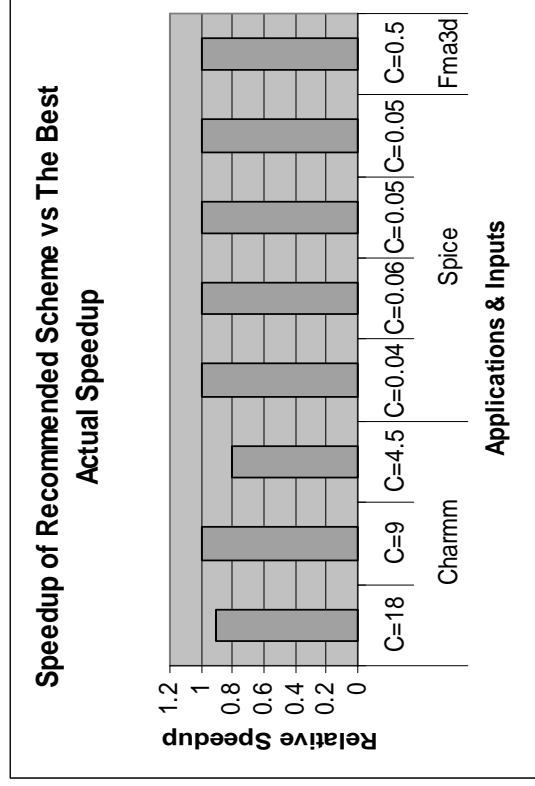
App	N	CON	MOB	OTH	CHR	CLUS	Recommended Scheme		Experimental Result
							Regression	Decision Tree	
Moldyn	49152	96	2	1	0.41	2	RLL	SEL	REP
	127776	31			0.36	2	RLL	SEL	REP
	210,912	6.8			0.33	2	SEL	SEL	RLL
	263,424	1.5			0.29	2	SEL	SEL	RLL
Spark98	90327	5	2	0	0.18	2	SEL	SEL	SEL
	21882	4.8			0.2	2	SEL	SEL	RLL



Experimental Results

Charmm, Spice & Fma3d

App	N	CON	MOB	OTH	CHR	CLUS	Recommended Scheme		Experimental Result
							Regression	Decision Tree	
Charmm	996,864	18	2	0	0.14	2	SEL	SEL	REP
	996,864	9			0.15	2	SEL	SEL	SEL
	1,993,734	4.5			0.13	2	SEL	SEL	RLL
Spice	186,943	0.04	28	0	0.13	2	SEL	SEL	SEL
	99,190	0.06			0.13	2	SEL	SEL	SEL
	89,925	0.05			0.13	2	SEL	SEL	SEL
	33,725	0.05			0.13	2	SEL	SEL	SEL
Fma3d	174,762	0.5	8	1	0.13	3	SEL	SEL	SEL



Conclusion

Contributions:

- A systematic process of constructing a performance model.
- Ability to predict and select the best available algorithms for reduction parallelization.

Future Work:

- To improve the accuracy of the prediction by refining
 - Pattern descriptor estimation
 - Parameterized synthetic loop
- To test the the accuracy of the prediction on more real loops.
- To extend current run-time adaptive approach to dynamic programs with large reusability.