Withall
=
Nested For + If

Sheena Mathew,
Div. of Computer Engg.,
School of Engineering,
CUSAT,
Kerala, India.

Tomsy Paul,
Dept. Of Computer Applications,
RIT Kottayam,
Kerala, India.

Overview

- Typical Array processing
  - Lower Triangle of a Matrix
    - For(i=0;i<N;i++)

      For(j=0;j<N;j++)

      If (i<=j) A[i,j] = A[i,j] * 0.2;
  - Withall Loop
    - withall (i<=j) A[i,j] = A[i,j] * 0.2;

withall : Syntax

- withall (*expression) statement*
- *statement* should contain at least one Array reference
- If there are many, all have same dimensions and size
- *expression* involves Index variables in Array reference
- Examples
  - withall (i==j) A[i,j] = 1;
  - withall (i+j == N+1) printf(“%d”,A[i,j]);
- *Expression* can also be (i,j) – Applied to all elements
  - withall (i,j) A[i,j] = 0;

Motivation

- *with* loop of SAC
- X = [1,2,3];
- with ([0] <= k < [2]) : 7

  modarray(x)
- X becomes [7,7,3]

Current Literature

- List Comprehension
- From Functional Programming (Python – numpy)
- Example (Haskell)
- [(i,j)|i<-[1..3],j<-[1..3],i<=j]
- Result
  - [(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)]
- Complexity - $O(N^2)$

Current Literature

- Boolean Indexing
- Present in MATLAB, Python, R, etc.
- a=[1,2,3,4,5]
- a>3
- Results in [0,0,0,1,1]
- a[a>3]=3
- print a
- Output [1,2,3,3,3]

Implementing withall

- Syntactic Sugar
- Efficient Implementation
  - NP Hard
  - Solution ?
- Restricted Grammar

Restricted Grammar

- withall_loop

-> withall withall_expression statement

| withall ID_LIST statement

;

- withall_expression

-> withall_expression && withall_expression

| withall_expression || withall_expression

| (withall_expression)

| !withall_expression

| withall_term

;

Restricted Grammar

- withall_term -> ID RELOP expression

;

- RELOP can be any of the six operators <,<=,!=,==,>= and > and ID is any index variable used in statement.
- Example
  - i <= 10
  - j <= i
  - *not* i+j < N+1

Implementation

- For One Dimensional Arrays
  - Using List of Ordered Pairs, for each withall_expression
  - Examples
    - int a[1:100]
      - i < 50 represented by {(1,49)}
      - i > 25 represented by {(26,100)}
      - i == 50 represented by {(50,50)}
      - i != 50 represented by {(1,49),(51,100)}

Implementation

- For One Dimensional Arrays
  - Using List of Ordered Pairs, for each withall_expression
  - Advantages
    - Logical And, Or and Negation can be implemented as Union, Intersection and Complement of Ordered Pairs
    - O(m+n) for And and Or
    - O(n) for Negation

Implementation

- int a[1:100]
  - (i < 50) && (i > 20)
    - {(1,49)} Intersection {(21,100)} ---> {(21,49)}
  - (i < 50) || (i > 75)
    - {(1,49)} Union {(76,100)} ---> {(1,49),(76,100)}
  - !((i < 30) && (i>20))
    - Complement {(21,29)} ---> {(1,20),(30,100)}

Implementation

- Language Used
- Cray Chapel
  - Free & Open Source
  - Excellent Support
  - LR Parser
  - Modules and Iterators
  - Version 1.14

Implementation

- Platform
  - Desktop
    - Intel Xeon 2 GHz, 6 core processor
    - 8 GB RAM

Implementation

- Procedure
  - Modified Chapel Compiler
  - Files modified
    - compiler/parser/chapel.lex
    - compiler/parser/chapel.ypp
    - compiler/AST/ForLoop.cpp
    - compiler/include/ForLoop.h
    - modules/internal/ChapelArray.chpl
    - modules/internal/ChapelIteratorSupport.chpl

Experiment

- Withall Vs Nested For + If
- Array Size (N)
  - 10^3
  - 10^4
  - 10^6
  - 10^8

Experiment

- Withall Vs Nested For + If
- withall_expressions

Number	withall_expression
1	i
2	i==N/2
3	i<=N/2
4	!(i>N/2)
5	i<=N/4 || i>=3*N/4
6	i>=N/4 && i<=N/2
7	!((i>=N/4 && i<=N/2) || (i>=3*N/4 && i<=N))

Results

- Comparison
  - Execution Time
  - Target Code Size
  - Compilation Time

Execution Time (Seconds)

withall_ expression		Array Size			
		10^3	10^4	10^6	10^8
1	**w**[a]	2.10×10^{-6}	1.93×10^{-5}	1.18×10^{-3}	1.50×10^{-1}
	f+i[b]	2.50×10^{-6}	1.50×10^{-5}	1.16×10^{-3}	1.61×10^{-1}
2	**w**	9.00×10^{-7}	1.00×10^{-6}	1.00×10^{-6}	1.80×10^{-6}
	f+i	2.50×10^{-6}	1.75×10^{-5}	1.68×10^{-3}	1.71×10^{-1}
3	**w**	1.80×10^{-6}	1.14×10^{-5}	6.04×10^{-4}	7.52×10^{-2}
	f+i	2.90×10^{-6}	1.96×10^{-5}	1.83×10^{-3}	1.84×10^{-1}
4	**w**	2.50×10^{-6}	1.19×10^{-5}	6.07×10^{-4}	7.44×10^{-2}
	f+i	2.70×10^{-6}	1.98×10^{-5}	1.83×10^{-3}	1.84×10^{-1}
5	**w**	3.40×10^{-6}	1.10×10^{-5}	5.71×10^{-4}	7.23×10^{-2}
	f+i	3.30×10^{-6}	2.64×10^{-5}	2.45×10^{-3}	2.45×10^{-1}
6	**w**	2.00×10^{-6}	7.00×10^{-6}	3.19×10^{-4}	3.75×10^{-2}
	f+i	3.30×10^{-6}	2.43×10^{-5}	2.36×10^{-3}	2.34×10^{-1}
7	**w**	5.30×10^{-6}	1.52×10^{-5}	5.96×10^{-4}	7.00×10^{-2}
	f+i	4.20×10^{-6}	3.31×10^{-5}	3.37×10^{-3}	3.13×10^{-1}

[a] **w** denotes **withall**

[b] **f+i** denotes for loop + if statement

Target Code Size (KB)

withall_ expression	Target Code Size	
	w	**f+i**
1	3667.89	3667.8
2	3668.04	3667.8
3	3668.04	3667.8
4	3668.08	3667.8
5	3672.37	3667.76
6	3672.27	3667.76
7	3672.40	3667.80

Time to Compile (Seconds)

withall_ expression		Array Size			
		10^3	10^4	10^6	10^8
1	**w**	10.19	10.68	10.59	10.26
	f+i	10.52	10.42	10.14	9.98
2	**w**	10.60	10.38	10.42	10.36
	f+i	10.41	10.19	10.17	10.18
3	**w**	10.29	10.49	10.40	10.36
	f+i	10.13	10.17	10.21	10.16
4	**w**	10.48	10.45	10.43	10.39
	f+i	10.43	10.23	10.14	10.13
5	**w**	10.85	10.96	10.35	10.71
	f+i	10.64	10.09	10.11	10.11
6	**w**	11.21	10.51	10.43	10.45
	f+i	10.63	10.44	10.10	10.11
7	**w**	10.57	10.60	10.18	10.56
	f+i	10.50	10.14	10.09	10.02

Conclusion

- Withall is concise
- More than a Syntactic Sugar
  - One Dimension – Tuple based
  - Much Faster than For + If
- Two Dimensional
  - List of Polygons
  - Polygon Intersection for And
  - Polygon Union for Or

Reference

SaC-Home, http://www.sac-home.org/doku.php

Grelck C.: Shared memory multiprocessor support for functional array processing in SAC. Journal of Functional Programming, vol. 15(3), 353–401 (2005).

Python 3.7.5rc1 documentation, https://docs.python.org/3/tutorial/datastructures.html

NumPy v1.18.dev0 Manual, https://numpy.org/devdocs/reference/arrays.indexing.html#boolean-array-indexing

Chapel: Productive Parallel Programming, https://chapel-lang.org/

Thank You!!