



# SWAMP: Smith-Waterman using Associative Massive Parallelism

**Shannon I. Steinfadt and Johnnie Baker**  
Parallel and Associative Computing Lab  
Computer Science Department  
Kent State University

9th International Workshop on  
Parallel and Distributed Scientific and  
Engineering Computing (PDSEC '08)



## Topics Overview

- **Sequence Alignment**
  - What type of data we're aligning
  - How we're trying to align it
  - Smith-Waterman example
- **Associative Parallel Model (ASC)**
- **SWAMP Algorithm**
- **Future Work**

## Sequence Alignment

Given two sequences:

DNA *nucleotides* {C,T,G,A}

Amino Acids {a, r, n, d, c, q, e, g, h, i, l, k, m, f, p, s, t, w, y, v}

Align them to find the longest, most  
common subsequence

ctcgccgcg cgcggacgct ccacgtgtcc cccgtctacc

gggccctcct ggctcccaac agcttctcag ttcccacttc

## Sequence Alignment

Given two sequences:

DNA *nucleotides* {C,T,G,A}

Amino Acids {a, r, n, d, c, q, e, g, h, i, l, k, m, f, p, s, t, w, y, v}

Align them to find the longest, most  
common subsequence

gcggacgct ccacg-tgtc--c --c- tcgcccgcgc cc-cgtctacc

||:|:| |::|-|:|:|--| --|-|:|:|:|:| |:-|:|

gggccct cctggctcccaac agc ttctcagttc ccacttc

Similar Characters  Similar Structure  
 Similar Function

KENT STATE  
UNIVERSITY

## Sequence Alignment

Similar Characters ⇔ Similar Structure  
⇔ Similar Function

↓

Ancestral Relationships  
Gene Functionality  
Aid in Drug Discovery

KENT STATE  
UNIVERSITY

## Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence  
characters against each other

		C	T	G	G
	0	0	0	0	0
C	0				
A	0				
T	0				
T	0				
G	0				

KENT STATE UNIVERSITY

### Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence characters against each other

		<b>C</b>	<b>T</b>	<b>G</b>	<b>G</b>
		↘	↓	0	0
<b>C</b>	→				
<b>A</b>	0				
<b>T</b>	0				
<b>T</b>	0				
<b>G</b>	0				

**Cost Key**  
 Match +10  
 Miss -3

Insert a Gap -3  
 Extend a Gap -1

KENT STATE UNIVERSITY

### Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence characters against each other

		<b>C</b>	<b>T</b>	<b>G</b>	<b>G</b>
		↘	↓	0	0
<b>C</b>	→	10			
<b>A</b>	0				
<b>T</b>	0				
<b>T</b>	0				
<b>G</b>	0				

**Cost Key**  
 Match +10  
 Miss -3

Insert a Gap -3  
 Extend a Gap -1

KENT STATE UNIVERSITY

### Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence characters against each other

	C	T	G	G
	0	0	0	0
C	0	10		
A	0			
T	0			
T	0			
G	0			

Cost Key  
 Match +10  
 Miss -3

Insert a Gap -3  
 Extend a Gap -1

KENT STATE UNIVERSITY

### Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence characters against each other

	C	T	G	G
	0	0	0	0
C	0	10	6	
A	0			
T	0			
T	0			
G	0			

Cost Key  
 Match +10  
 Miss -3

Insert a Gap -3  
 Extend a Gap -1

KENT STATE UNIVERSITY

### Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence characters against each other

		C	T	G	G
		0	0	0	0
C		0	10	6	5
A		0			
T		0			
T		0			
G		0			

Cost Key  
 Match +10  
 Miss -3

Insert a Gap -3  
 Extend a Gap -1

KENT STATE UNIVERSITY

### Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence characters against each other

		C	T	G	G
		0	0	0	0
C		0	10	6	5
A		0			
T		0			
T		0			
G		0			

Cost Key  
 Match +10  
 Miss -3

Insert a Gap -3  
 Extend a Gap -1

## Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence  
characters against each other

		C	T	G	G
		0	0	0	0
C		0	10	6	5
A		0			
T		0			
T		0			
G		0			

Cost Key  
Match +10  
Miss -3

Insert a Gap -3  
Extend a Gap -1

## Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence  
characters against each other

		C	T	G	G
		0	0	0	0
C		0	10	6	
A		0	6		
T		0			
T		0			
G		0			

Cost Key  
Match +10  
Miss -3

Insert a Gap -3  
Extend a Gap -1

## Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence  
characters against each other

		C	T	G	G
		0	0	0	0
C		0	10	6	5
A		0	6	7	
T		0	5		
T		0			
G		0			

Cost Key  
Match +10  
Miss -3

Insert a Gap -3  
Extend a Gap -1

## Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence  
characters against each other

		C	T	G	G
		0	0	0	0
C		0	10	6	5
A		0	6	7	3
T		0	5	16	
T		0	4		
G		0			

Cost Key  
Match +10  
Miss -3

Insert a Gap -3  
Extend a Gap -1



## Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence  
characters against each other

		C	T	G	G
		0	0	0	0
C		0	10	6	5
A		0	6	7	3
T		0	5	16	12
T		0	4	15	
G		0	3		

Cost Key  
Match +10  
Miss -3

Insert a Gap -3  
Extend a Gap -1

## Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence  
characters against each other

		C	T	G	G
		0	0	0	0
C		0	10	6	5
A		0	6	7	3
T		0	5	16	12
T		0	4	15	13
G		0	3	11	

Cost Key  
Match +10  
Miss -3

Insert a Gap -3  
Extend a Gap -1

## Aligning using Smith-Waterman Algorithm

Compare all possible combinations of sequence  
characters against each other

		C	T	G	G	
		0	0	0	0	
C		0	10	6	5	4
A		0	6	7	3	2
T		0	5	16	12	11
T		0	4	15	13	10
G		0	3	11	25	

Cost Key  
Match +10  
Miss -3

Insert a Gap -3  
Extend a Gap -1

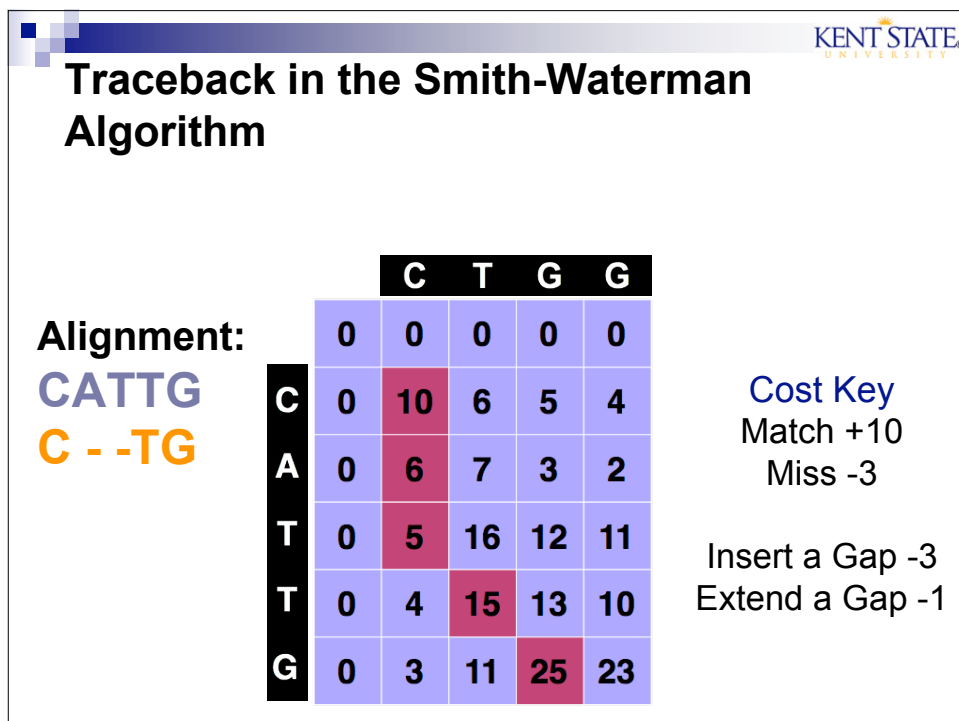
## Aligning using Smith-Waterman Algorithm

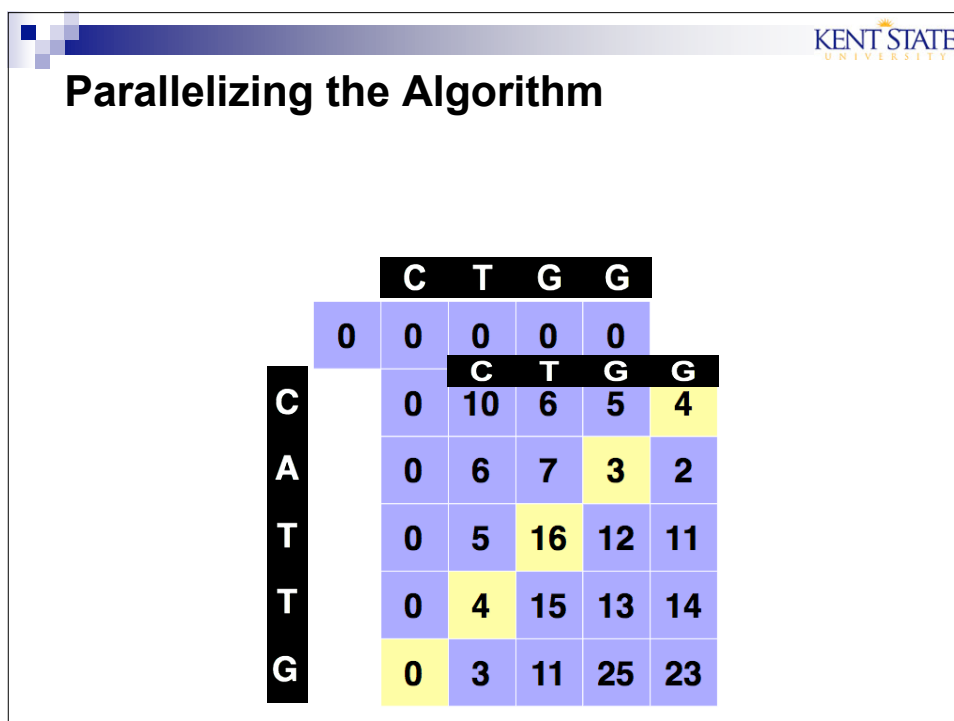
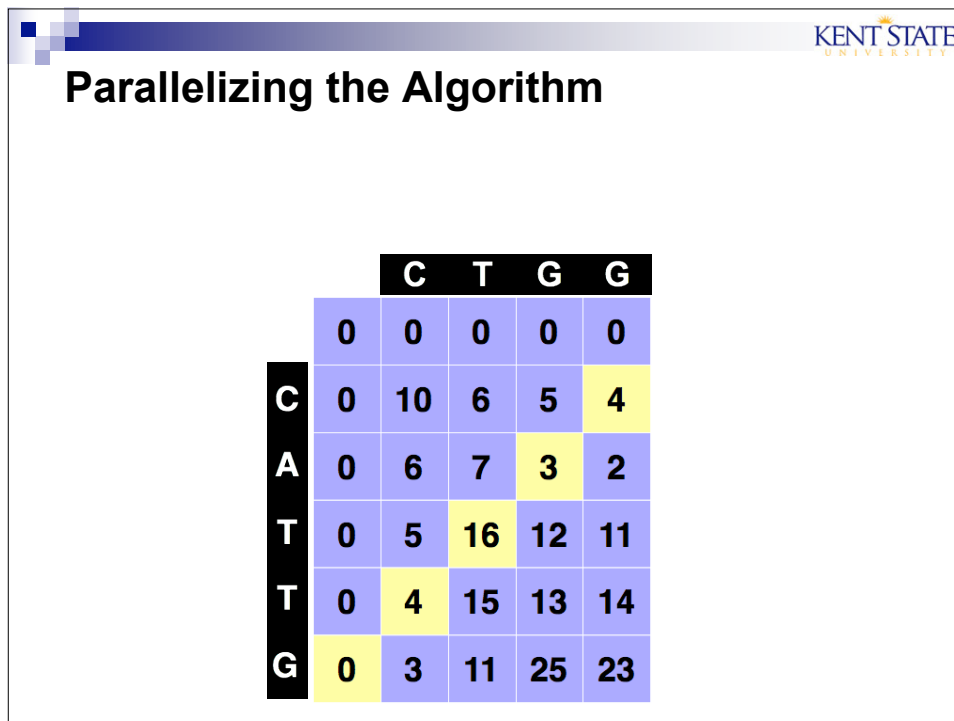
Compare all possible combinations of sequence  
characters against each other

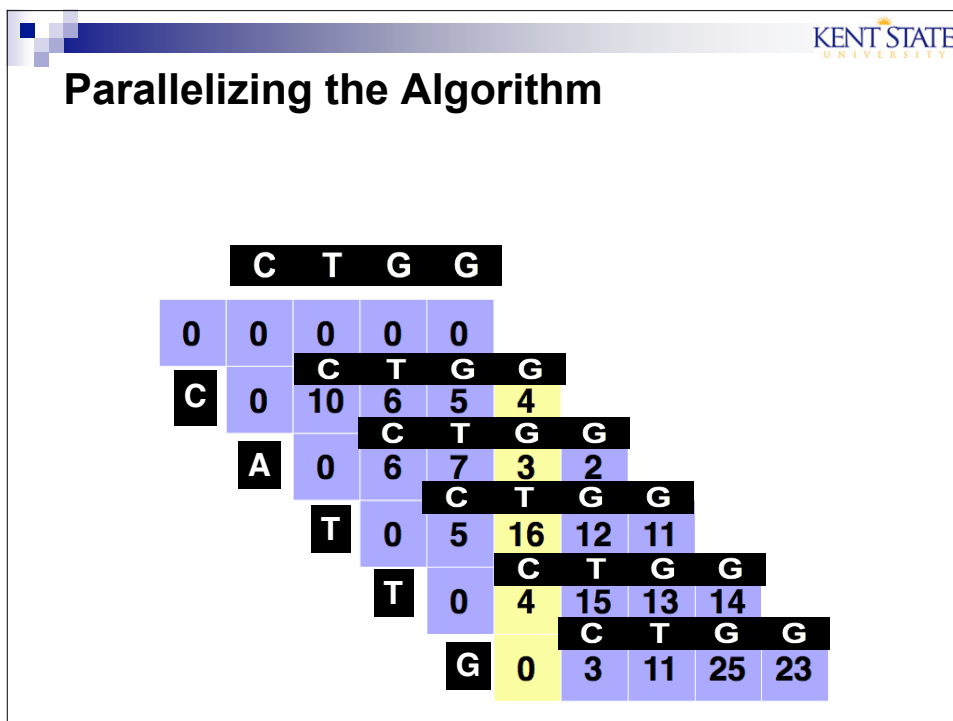
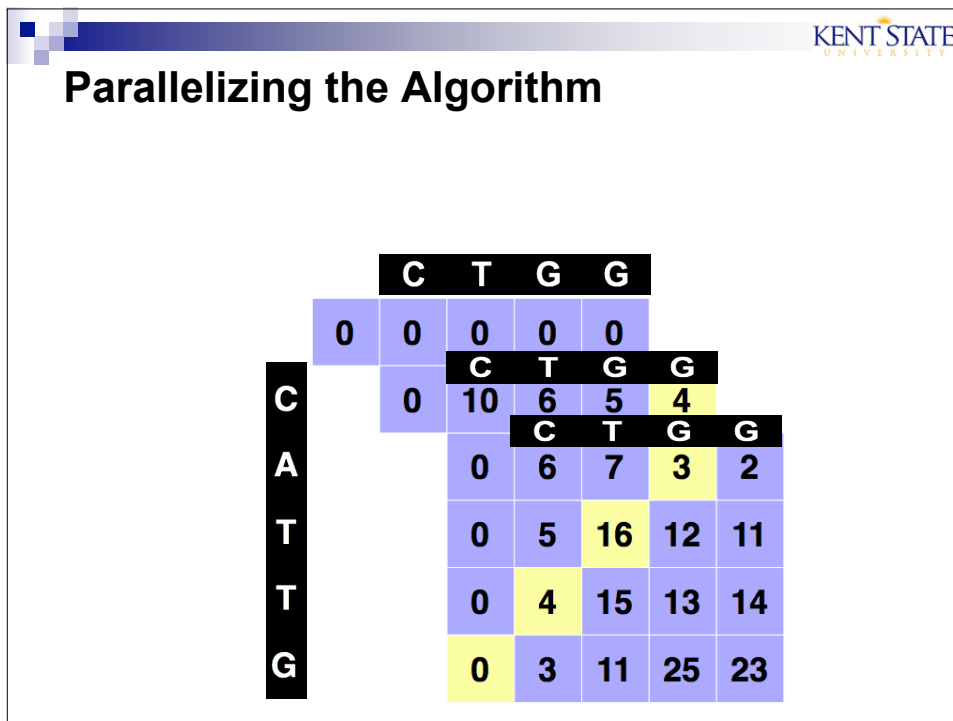
		C	T	G	G	
		0	0	0	0	
C		0	10	6	5	4
A		0	6	7	3	2
T		0	5	16	12	11
T		0	4	15	13	10
G		0	3	11	25	23

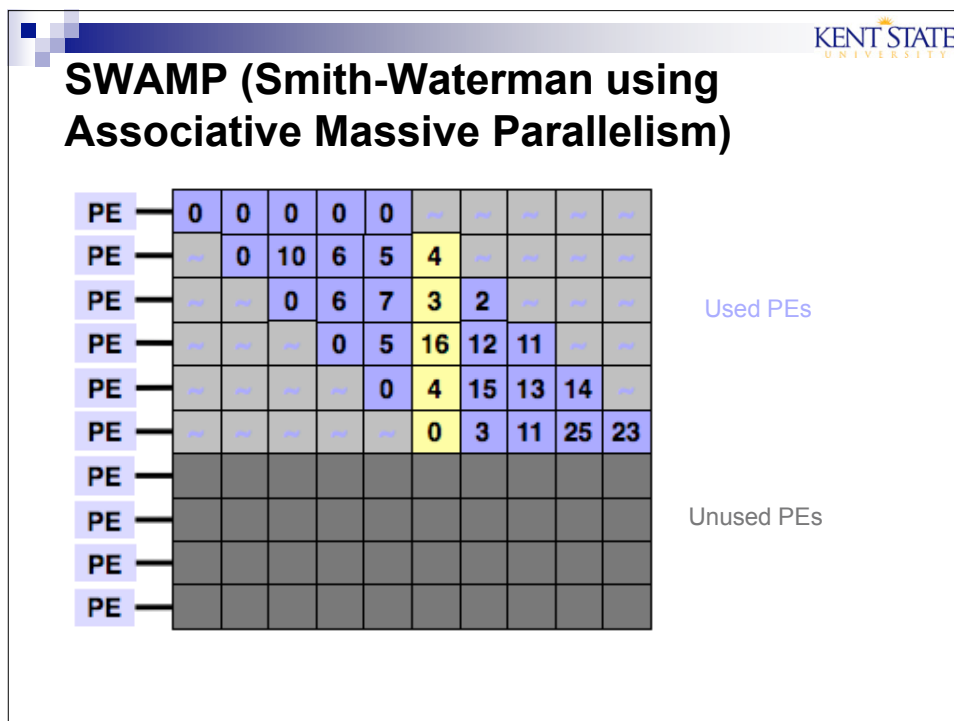
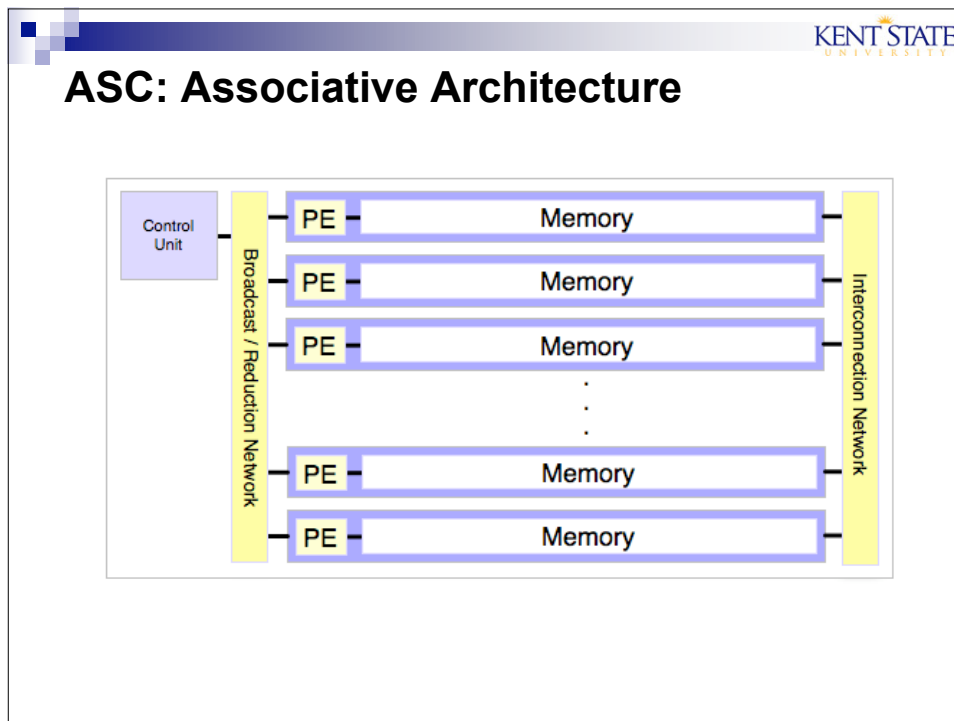
Cost Key  
Match +10  
Miss -3

Insert a Gap -3  
Extend a Gap -1









## ASC Features

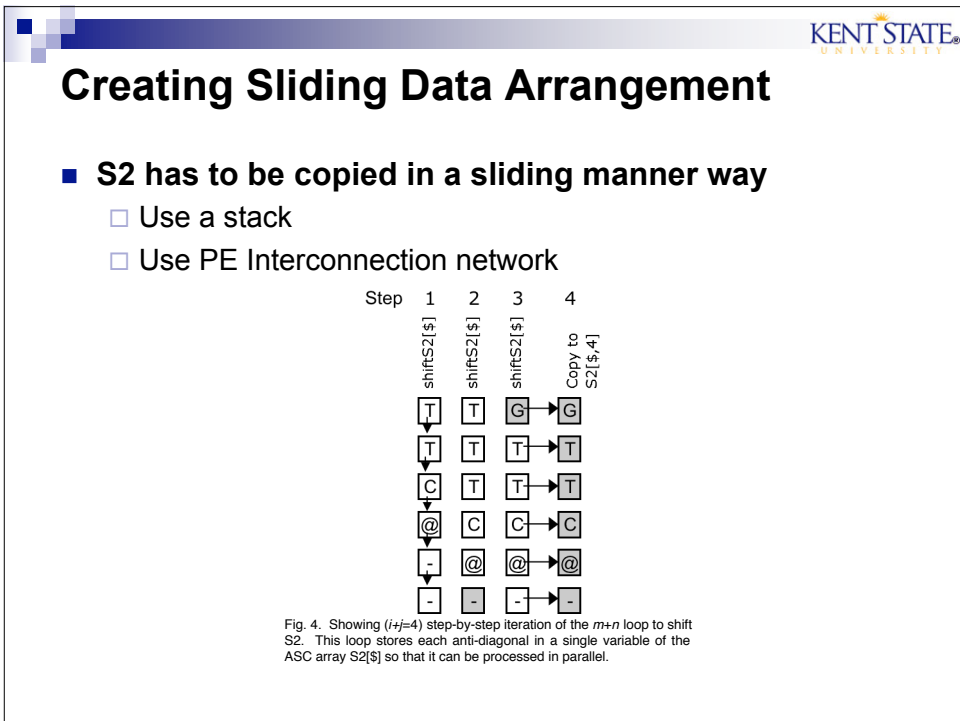
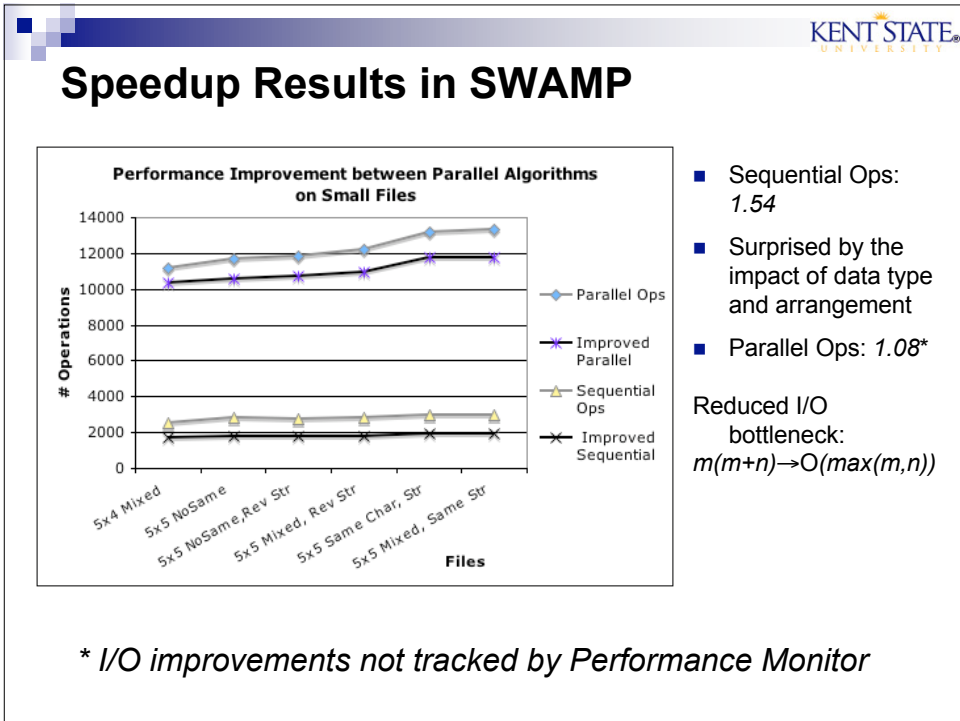
- **SIMD with special associative features**
- **Designed for fast associative searches**
  - Search based on content, not memory address

### Very fast operations for:

- **Finding Maximum / Minimum**
- **Finding if there are “Any Responders”**
- **“Pick One” active PE**

## ASC Advantages

- **Quick data movement in SIMD**
  - Move raw data in parallel
  - At each step, PEs follow the algorithmic steps for data movement in lock step
- **No message passing like MPI/PVM**
  - No store/forward
  - No headers
  - No explicit synchronizing





## SWAMP Analysis

$|S1| = m$  and  $|S2| = n$  (and  $m \geq n$ )

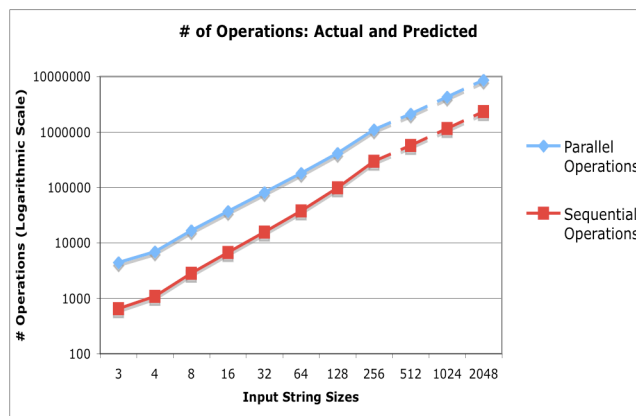
### ■ Sequential Smith-Waterman (Gotoh)

- $O(m*n)$  time,  $m*n$  space
- When  $|S1| = |S2|$ , it becomes an  $O(n^2)$  algorithm

### ■ SWAMP parallel algorithm

- Computation takes  $O(m+n)$  time with  $m+1$  PEs
- If actual number of PEs  $< m+1$ , assign  $\{(m+1) / \# \text{ PEs}\}$  work to each PE
  - 400 matrix elements / 100 PEs  $\Rightarrow$  each PE gets 4x the work

## Performance Measurements



- Based on actual measurements using ASC language and emulator
- Predictions shown with the dashed line

*Predictions calculated using linear regression and the least squares method*

## Current Work

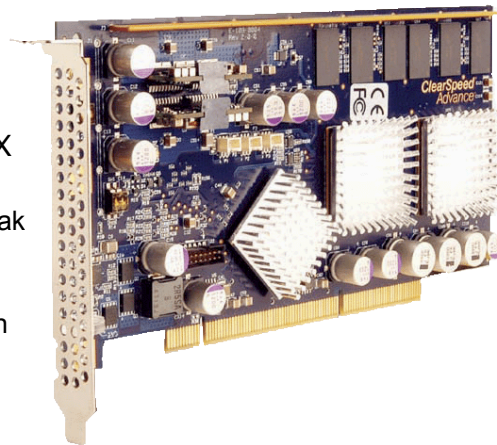
- Create a parallel ASC-language traceback
- Parallelize the data conversion (tilt) of the matrix to run more efficiently
- Use FASTA formatted files

## Current Work

- Extend the enhanced features of ASC to commercially available hardware

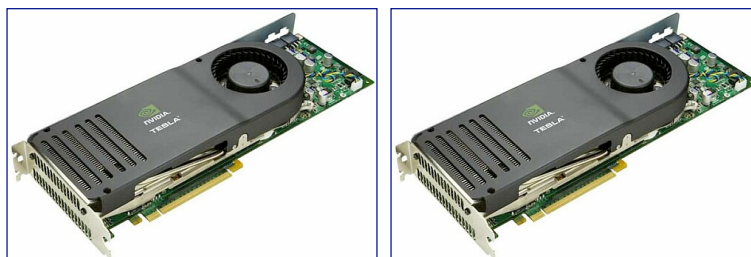
ClearSpeed  
Advance 620 PCI-X  
board

- 50 GFLOPS peak performance
- 25W average power dissipation



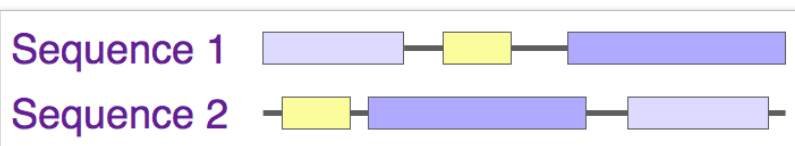
## Current Work

- **Extend the enhanced features of ASC to commercially available hardware**
  - Two NVIDIA Tesla
    - 518 Peak GFLOPS on Tesla Series
    - 170W peak, 120W typical



## Future Work

- **Work on extending SWAMP to SWAMP+, returning multiple non-overlapping sequences during the traceback**



## Questions ?

**Contact Info:**

**Shannon Steinfadt**

[ssteinfa@cs.kent.edu](mailto:ssteinfa@cs.kent.edu)

<http://www.cs.kent.edu/~ssteinfa>