AAlign: A SIMD Framework for Pairwise Sequence Alignment on x86-based Multi- and Many-core Processors

Kaixi Hou, Hao Wang, Wu-chun Feng
{kaixihou,hwang121,wfeng}@vt.edu

VirginiaTech Invent the Future

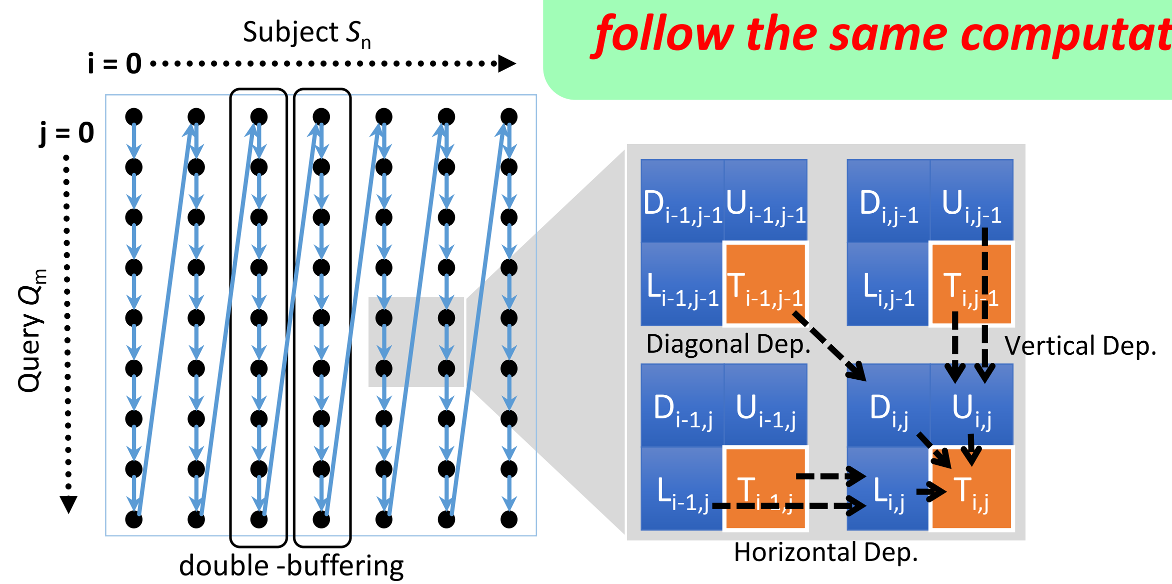
SyNeRG synergy.cs.vt.edu

Pairwise Sequence Alignment Algorithms

- Essential computational kernels in bioinformatics apps
  - Quantify similarity between pairs of sequences
- Different types of algorithms
  - Local alignment, e.g., Smith-Waterman
  - Global alignment, e.g., Needleman-Wunsch
- Different gap systems
  - Constant, linear, affine gap, etc.

Pairwise Sequence Alignment Algorithms

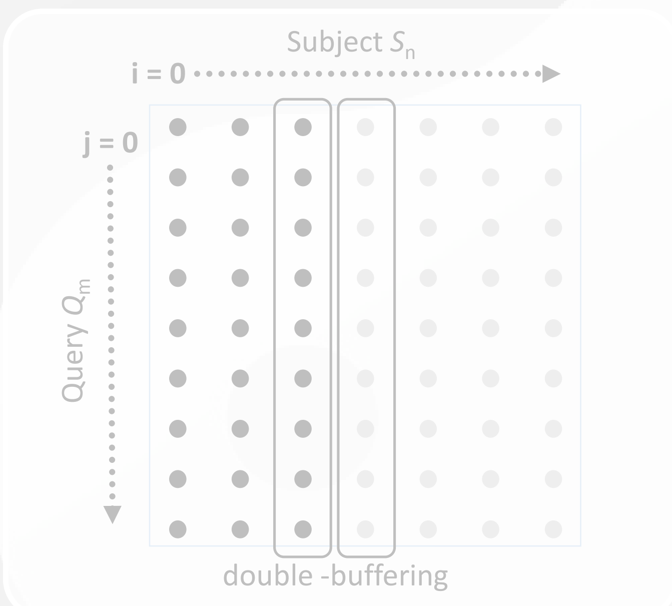
- Essential computational kernels in bioinformatics apps
  - Quantify similarity between pairs of sequences
- Different types of algorithms
  - Local alignment, e.g., Smith-Waterman
  - Global alignment, e.g., Needleman-Wunsch
- Different gap systems
  -

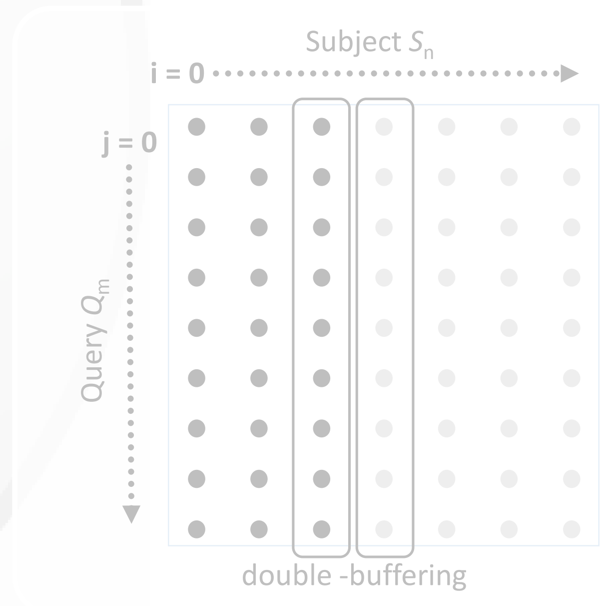


These algorithms and gap systems ALL follow the same computational pattern.

Pairwise Sequence Alignment Algorithms

- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods



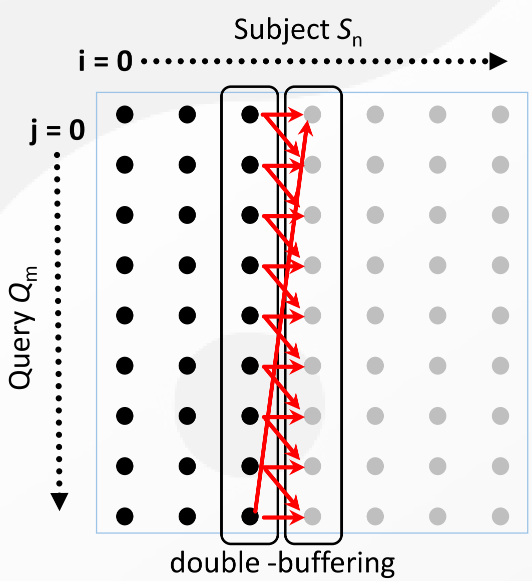


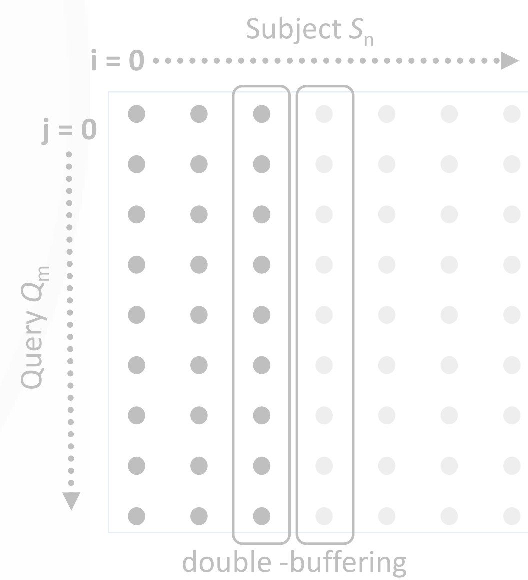
* M. Farrar, "Striped Smith-Waterman Speeds Database Searches …," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm …," J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results





1. Preprocess

VirginiaTech Invent the Future

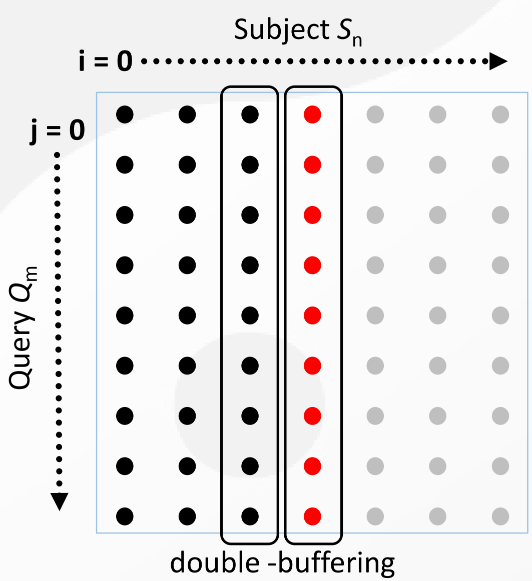
* M. Farrar, “Striped Smith-Waterman Speeds Database Searches … ,” Bioinformatics, 2007.

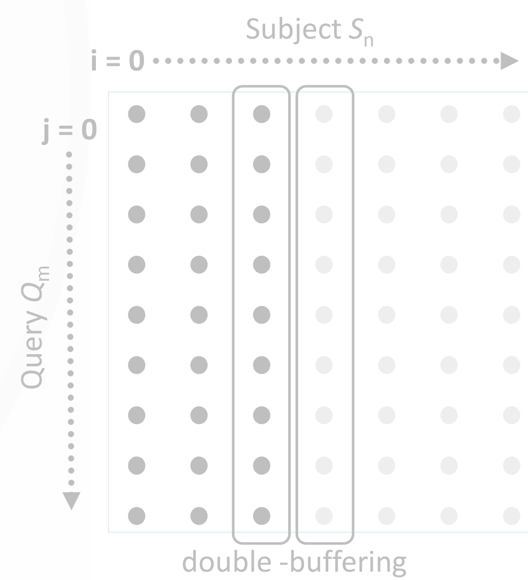
** A. Khajeh-Saeed et al., “Acceleration of the Smith- Waterman Algorithm … ,” J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results





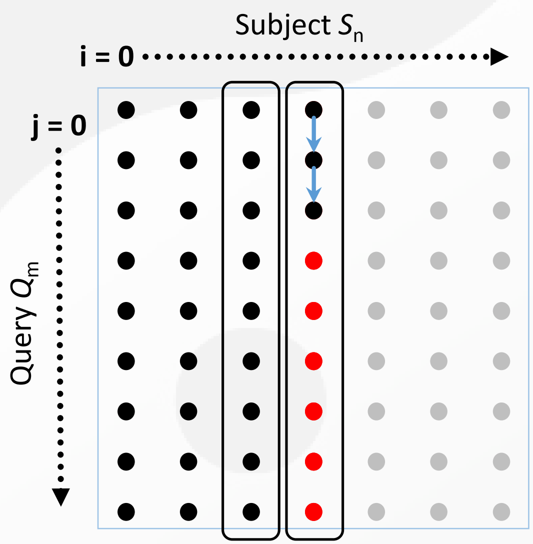
1. Preprocess **2. Check**

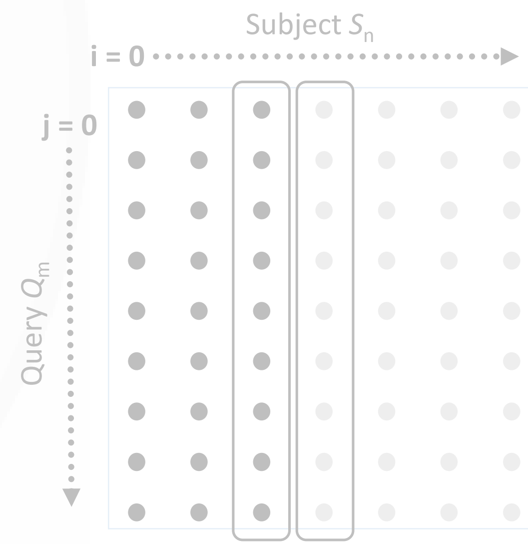
* M. Farrar, "Striped Smith-Waterman Speeds Database Searches …," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm …," J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results





1. Preprocess **2. Check** **3. Correct**

Virginia Tech Invent the Future

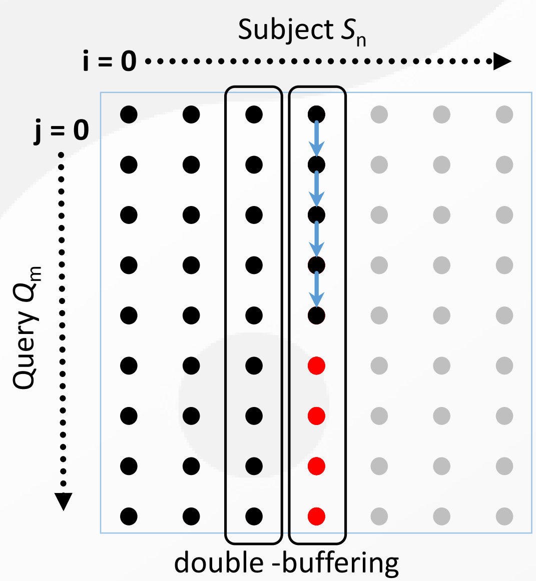
* M. Farrar, "Striped Smith-Waterman Speeds Database Searches ... ," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm ... ," J. Comp. Phys., 2010.

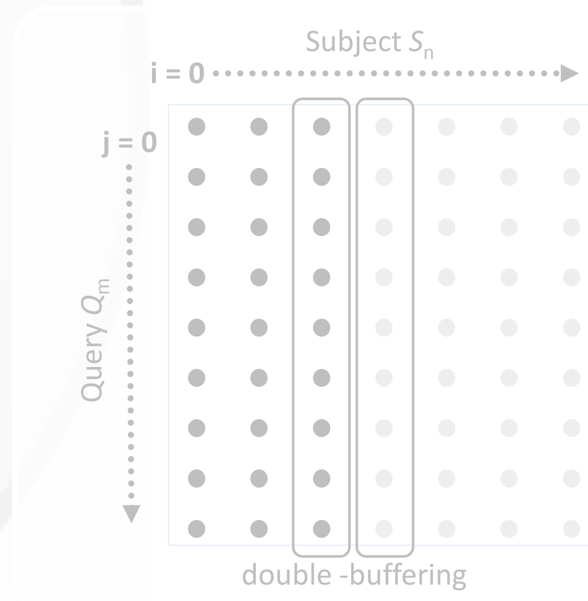
SyNeRG synergy.cs.vt.edu

Pairwise Sequence Alignment Algorithms

- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results





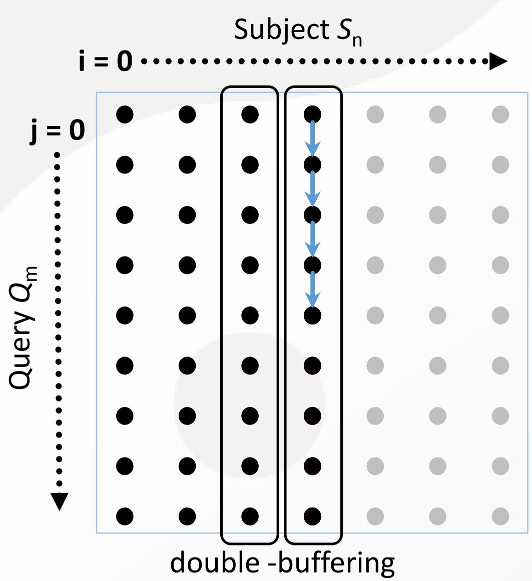
1. Preprocess **2. Check** **3. Correct**

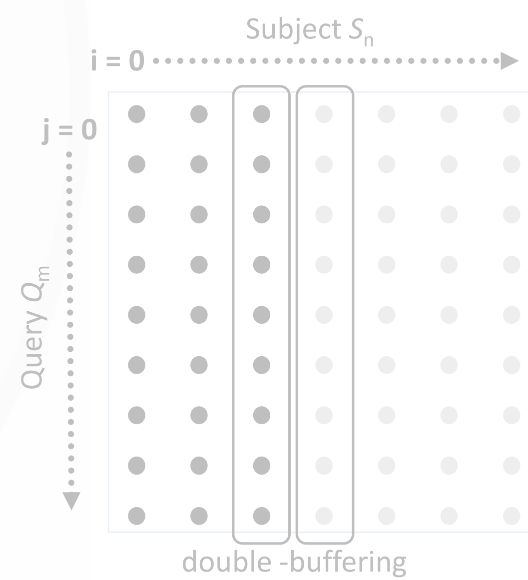
* M. Farrar, "Striped Smith-Waterman Speeds Database Searches … ," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm … ," J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results





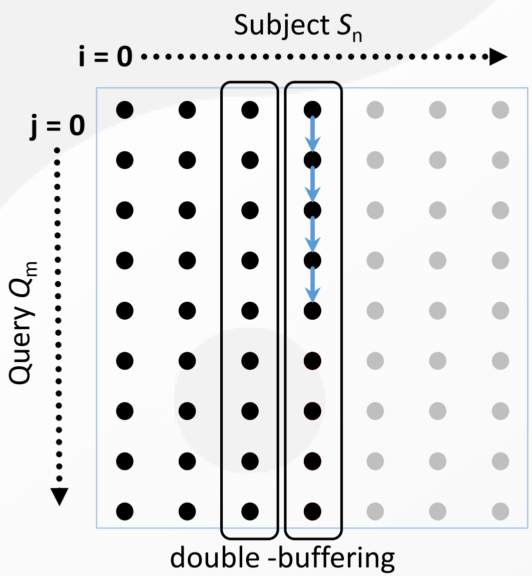
1. Preprocess **2. Check** **3. Correct**

* M. Farrar, "Striped Smith-Waterman Speeds Database Searches …," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm …," J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

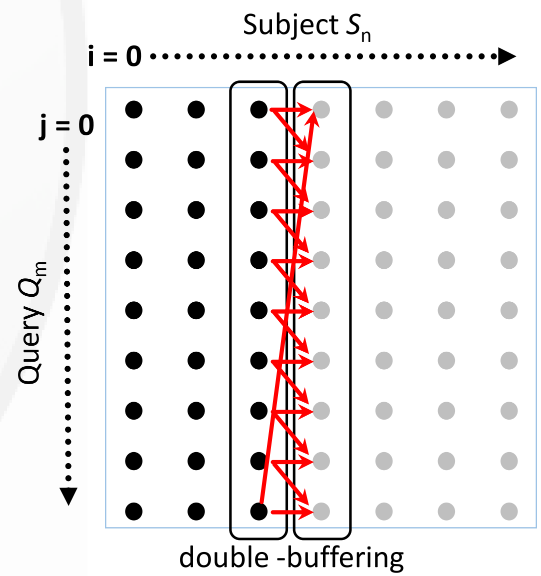
- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results



1. Preprocess **2. Check** **3. Correct**

Scan:** use a round of scan to validate results



1. Preprocess

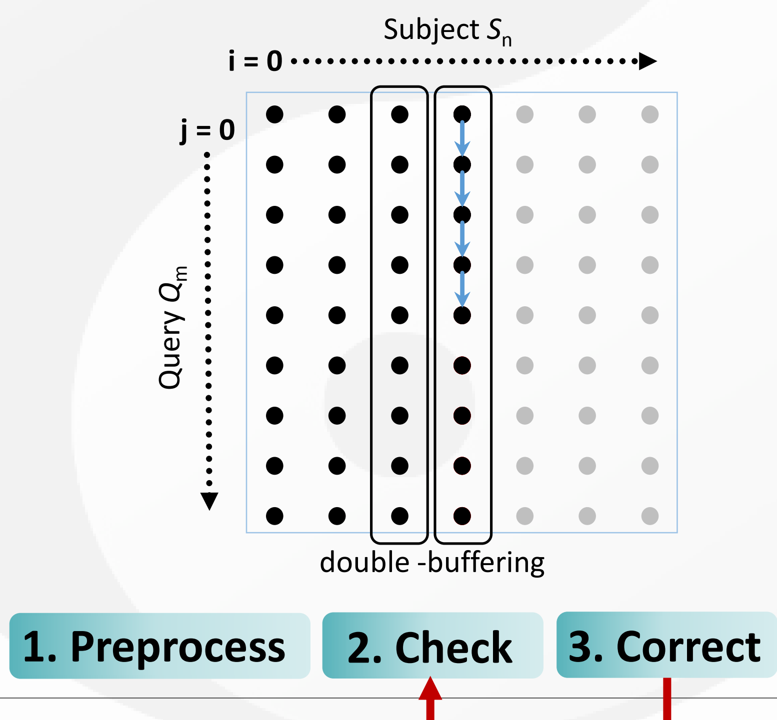
* M. Farrar, "Striped Smith-Waterman Speeds Database Searches … ," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm … ," J. Comp. Phys., 2010.

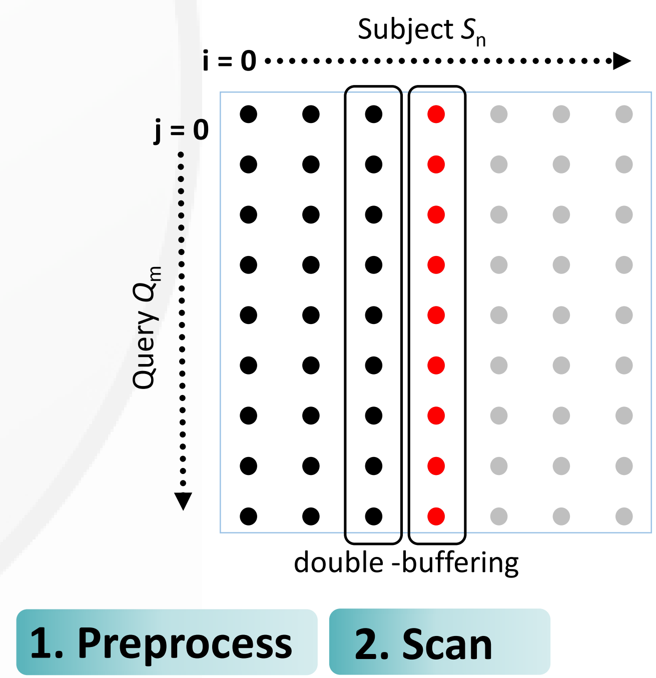
Pairwise Sequence Alignment Algorithms

- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results

Scan:** use a round of scan to validate results



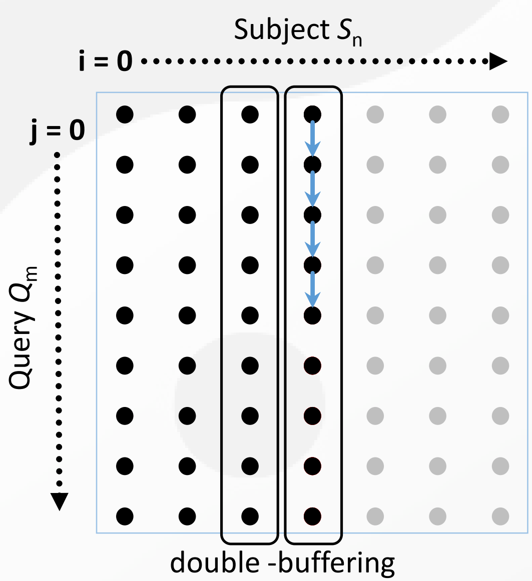


* M. Farrar, "Striped Smith-Waterman Speeds Database Searches … ," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm … ," J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

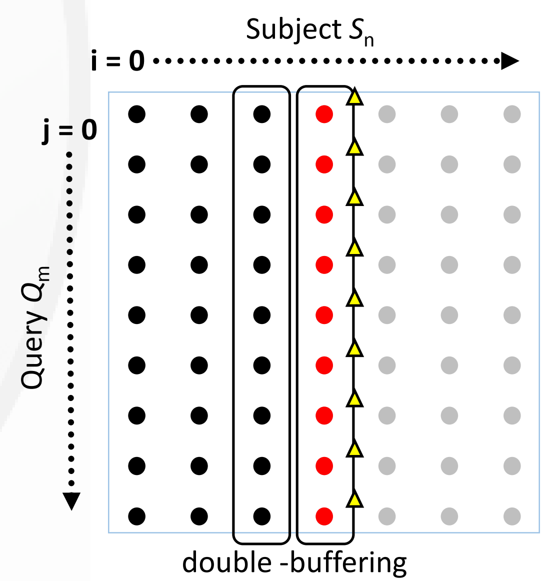
- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results



1. Preprocess **2. Check** **3. Correct**

Scan:** use a round of scan to validate results



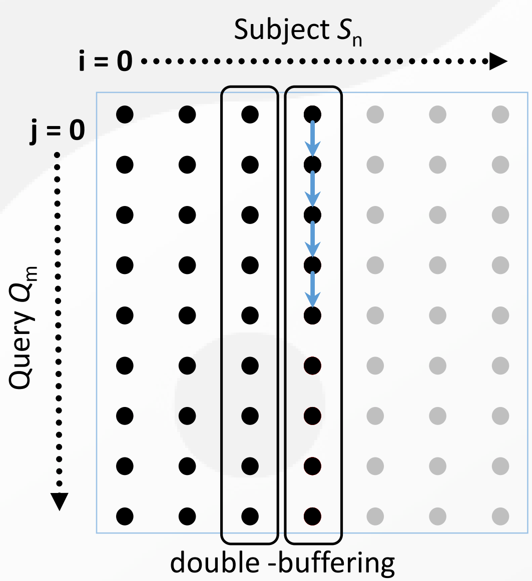
1. Preprocess **2. Scan**

* M. Farrar, "Striped Smith-Waterman Speeds Database Searches …," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm …," J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

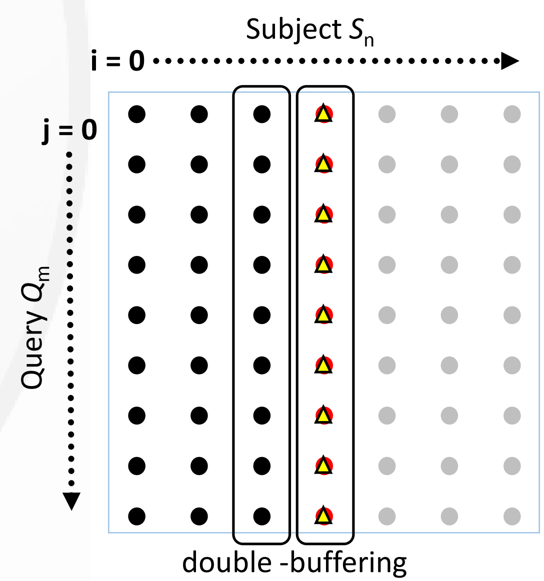
- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results



1. Preprocess | **2. Check** | **3. Correct**

Scan:** use a round of scan to validate results



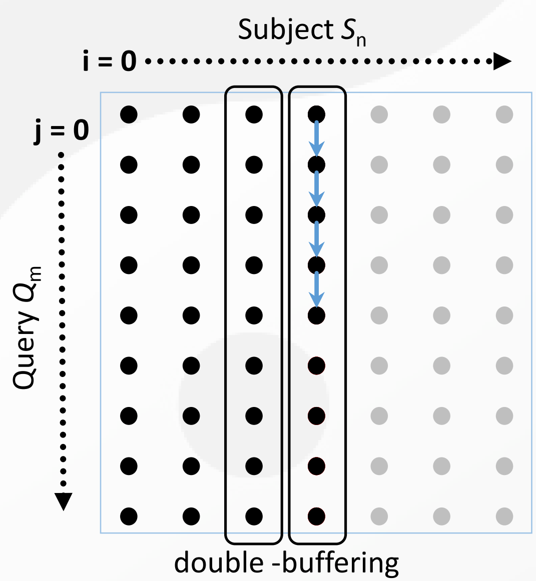
1. Preprocess | **2. Scan**

* M. Farrar, "Striped Smith-Waterman Speeds Database Searches …," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm …," J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

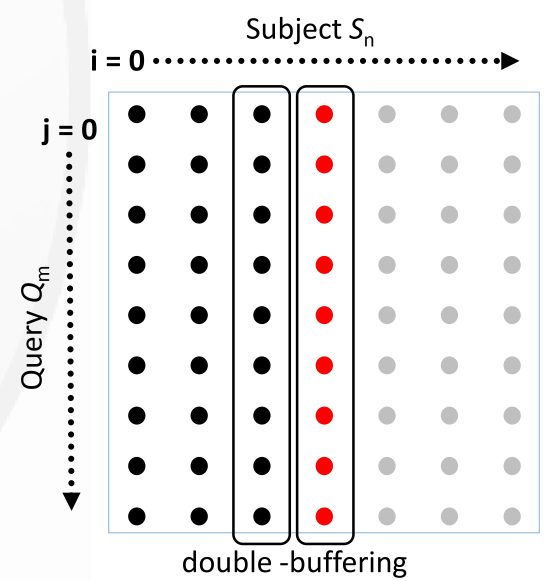
- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results



1. Preprocess | **2. Check** | **3. Correct**

Scan:** use a round of scan to validate results



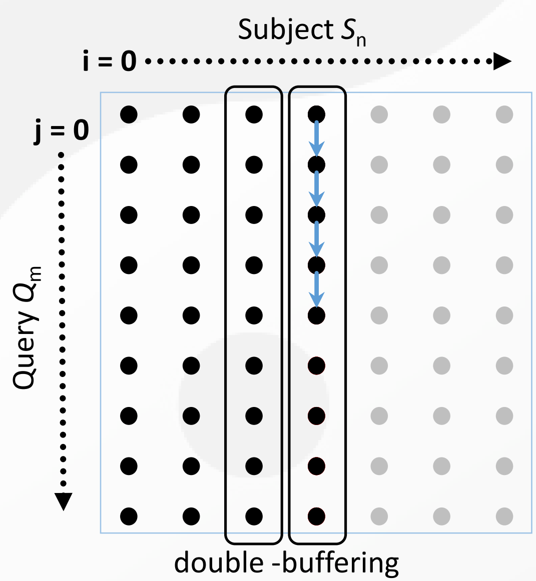
1. Preprocess | **2. Scan** | **3. Correct**

* M. Farrar, "Striped Smith-Waterman Speeds Database Searches … ," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm … ," J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

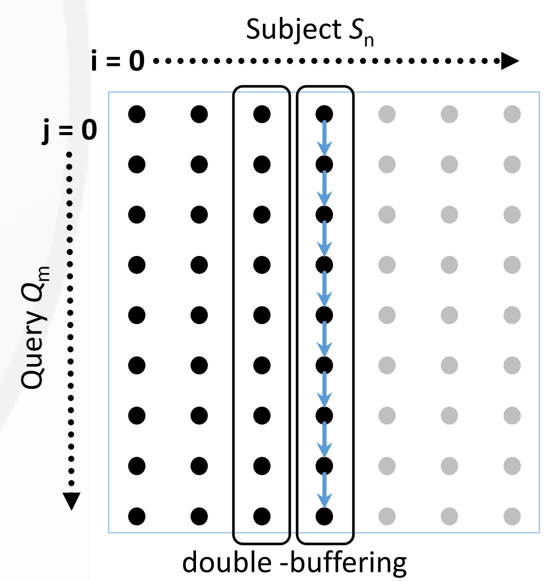
- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

Iterate*: use a certain number of iterations to validate results



1. Preprocess | 2. Check | 3. Correct

Scan:** use a round of scan to validate results



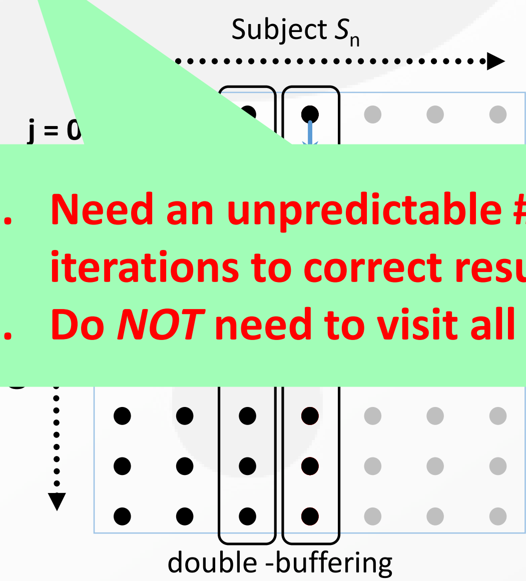
1. Preprocess | 2. Scan | 3. Correct

* M. Farrar, "Striped Smith-Waterman Speeds Database Searches …," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm …," J. Comp. Phys., 2010.

Pairwise Sequence Alignment Algorithms

- Different Vectorization Strategies
  - Two popular strategies: iterate & scan methods

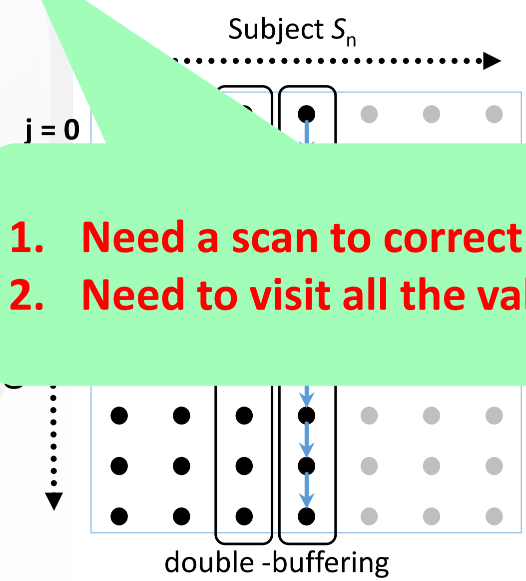
Iterate*: use a certain number of iterations to validate results



1. **Need an unpredictable # of iterations to correct results**
2. **Do *NOT* need to visit all the values**

1. Preprocess | **2. Check** | **3. Correct**

Scan:** use a round of scan to validate results



1. **Need a scan to correct results**
2. **Need to visit all the values**

1. Preprocess | **2. Scan** | **3. Correct**

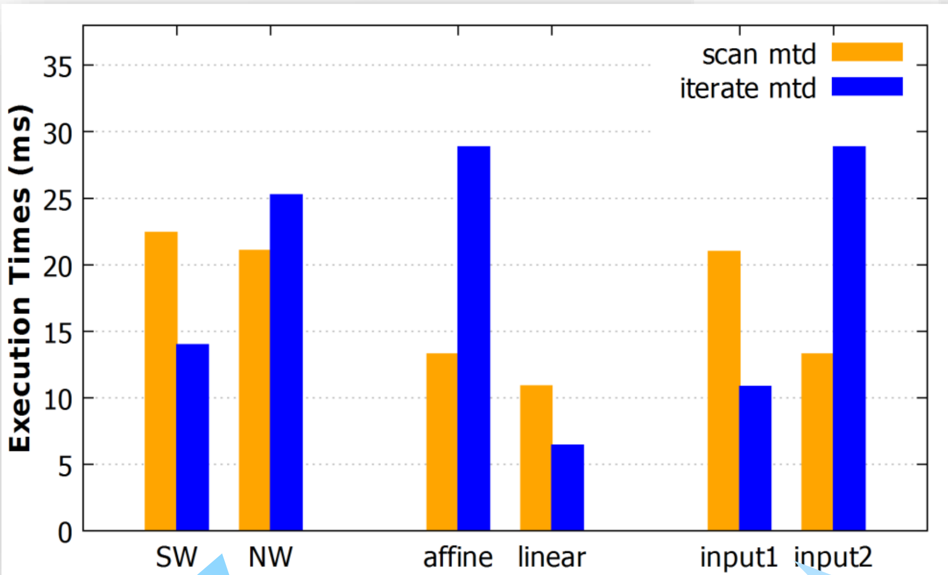
VirginiaTech Invent the Future

* M. Farrar, "Striped Smith-Waterman Speeds Database Searches …," Bioinformatics, 2007.
** A. Khajeh-Saeed et al., "Acceleration of the Smith- Waterman Algorithm …," J. Comp. Phys., 2010.

SyNeRG synergy.cs.vt.edu

Motivation & Challenges

- Which vectorization strategy is better on x86 systems?
  - Affected by different algorithms, configurations, inputs & platforms



Different Algorithm with same affine gap over same input data

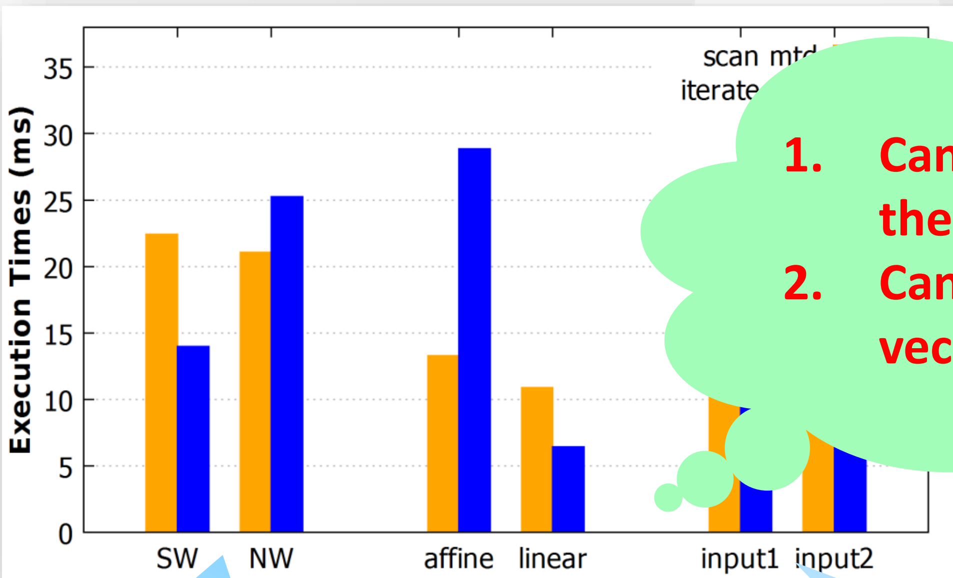
Different gap systems with same SW algorithm over same input data

Different inputs for the same SW algorithm with the same affine gap

kaixihou@vt.edu
IEEE IPDPS, 5/25/2016

Motivation & Challenges

- Which vectorization strategy is better on x86 systems?
  - Affected by different algorithms, configurations, inputs & platforms



1. **Can we automatically generate the vector codes?**
2. **Can we design an even better vectorization strategy?**

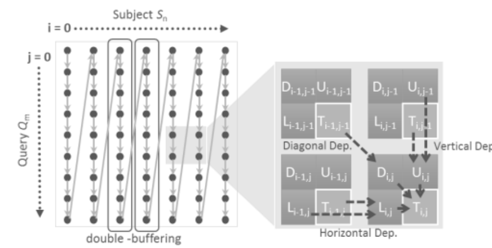
Different Algorithm with same affine gap over same input data

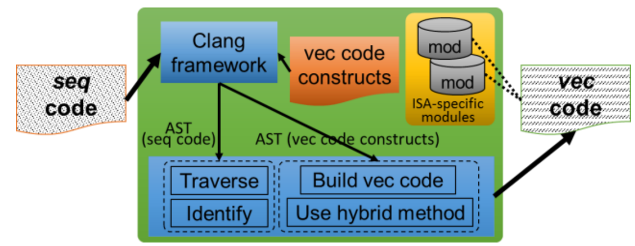
Different gap systems with same SW algorithm over same input data

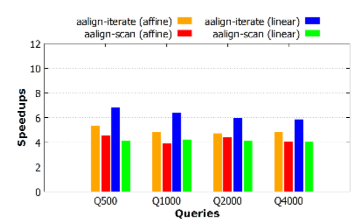
Different inputs for the same SW algorithm with the same affine gap

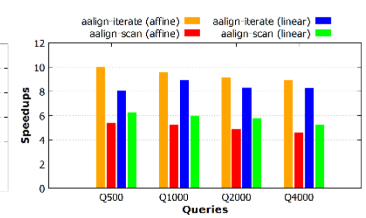
Roadmap

- Introduction & Motivation
- Background
  - Vector ISA
- AAlign Framework
  - Generalized Paradigm
  - Vector Modules
  - Vector Code Constructs
  - Hybrid Method
- Evaluation & Discussion
- Conclusion









Background: Vector ISA

- Vector Processing Units
  - Carry out a single operation over a vector of data simultaneously
- AVX2 Instructions
  - *Platform:* Vector ISA in current multi-core Haswell CPUs
  - *Width:* 256 bits (two 128-bit lanes)
  - *Operations:* Gather, cross-lane permute, per-element shift, etc.
- IMCI Instructions
  - *Platform:* Vector ISA in many-core Knights Corner MIC
  - *Width:* 512 bits (four 128-bit lanes)
  - *Operations:* Scatter, gather, inner/cross-lane permute, etc.

Approaches to Using Vector ISA

- Compiler-based approaches
  - Compiler options
  - Pragma directives

Approaches to Using Vector ISA

- Compiler-based appr[...]
  - Compiler options
  - Pragma directives

Issue:
Fail to auto-vec loops due to complex memory access, convoluted data rearrangement, etc.

kaixihou@vt.edu

Approaches to Using Vector ISA

- Compiler-based approaches
  - Compiler options
  - Pragma directives
- Manual optimization via …
  - Compiler intrinsics
  - Assembly code

Issue:
Tedious and error-prone.

Approaches to Using Vector ISA

- Compiler-based approaches
  - Compiler options
  - Pragma directives
- Manual optimization via …

Need a framework to hide the details of vector codes and make it cross-platform.

  - Assembly code

Serial C codes

```
for(i=0; i<2w; i++)
{
 if(i<offset)
  array[i]=x;
 else
  array[i]= array[i-offset];
}
```

Right-shift an array of length 2w

AVX2 intrinsics on CPUs

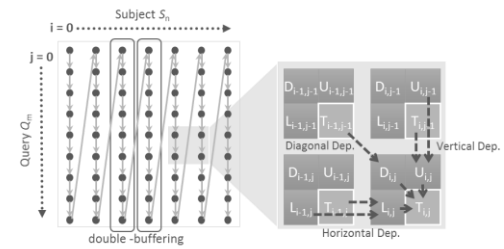
```
__m256i v_ret;
__m256i cv_rev = _mm256_set_epi32(6, 5, 4, 3, 2, 1, 0, 7);
v_ret = _mm256_load_si256((__m256i *)array);
v_ret = _mm256_permutevar8x32_epi32(v_ret, cv_rev);
v_ret = _mm256_insert_epi32(v_ret, x, 0);
```

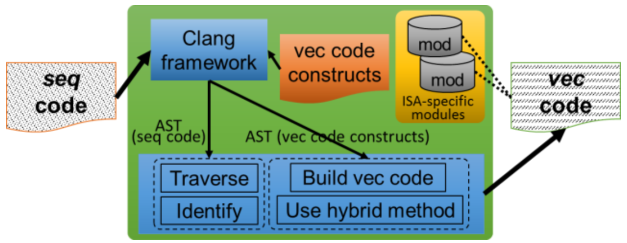
IMCI intrinsics on MIC

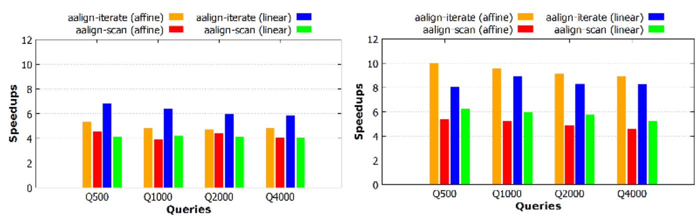
```
__m512i v_ret;
__m512i cv_rev = _mm512_set_epi32(14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,15);
unsigned short mask = 0xffff;    mask <<= num;
__m512i cv_fil = _mm512_set1_epi32(x);
v_ret = _mm512_load_epi32(array);
v_ret = _mm512_permutevar_epi32(cv_rev, v_ret);
v_ret = _mm512_mask_swizzle_epi32(cv_fil, mask, v_ret, _MM_SWIZ_REG_NONE);
```

Roadmap

- Introduction & Motivation
- Background
  - Vector ISA
- AAlign Framework
  - Generalized Paradigm
  - Vector Modules
  - Vector Code Constructs
  - Hybrid Method
- Evaluation & Discussion
- Conclusion



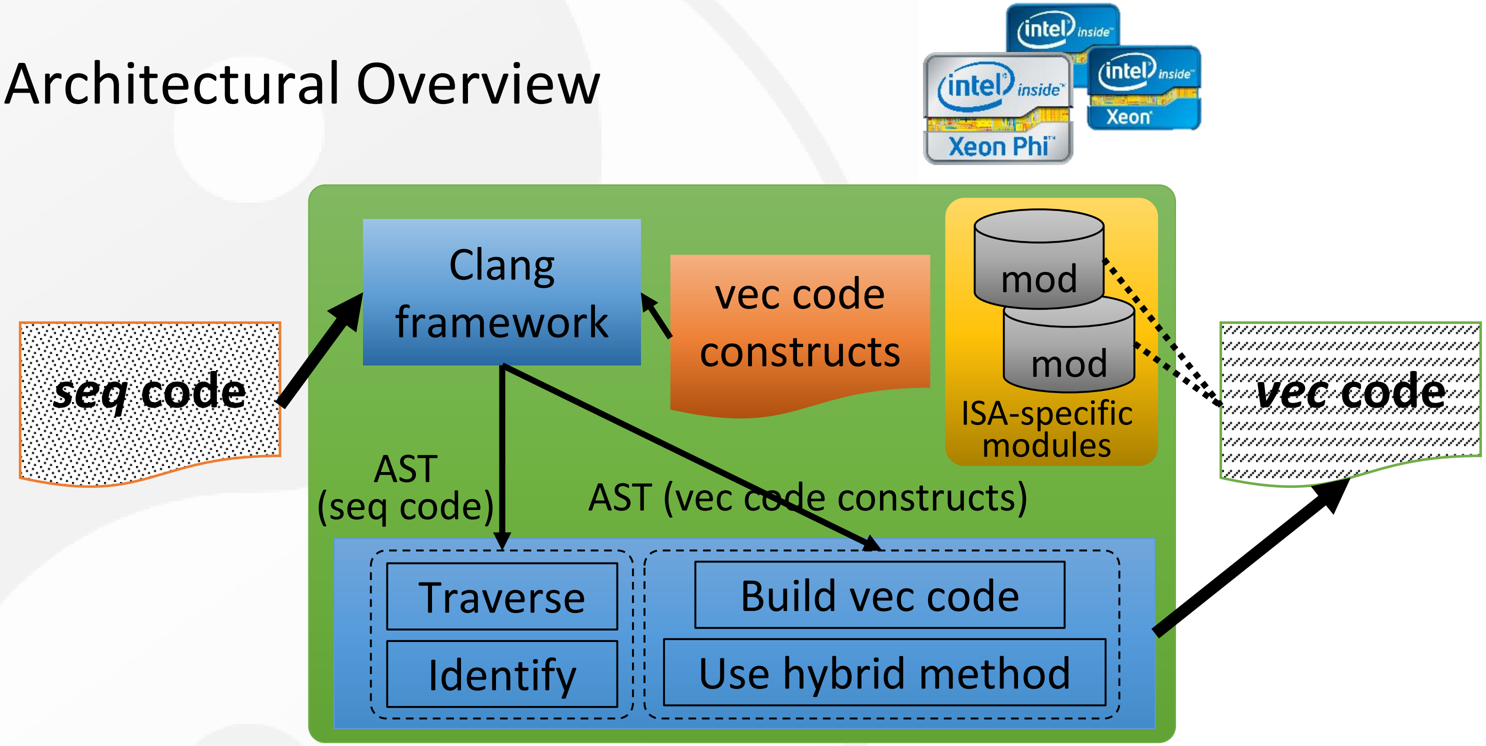


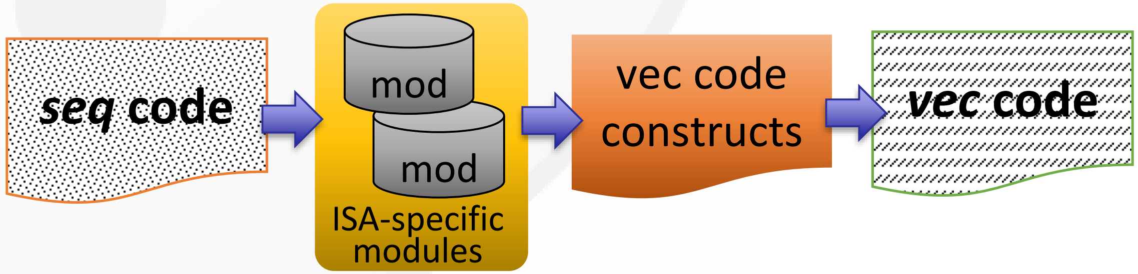


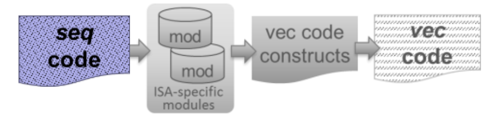
VirginiaTech Invent the Future
kaixihou@vt.edu
IEEE IPDPS, 5/25/2016
SyNeRGy synergy.cs.vt.edu

AAlign Framework

- Architectural Overview







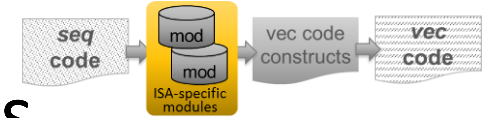
Proposed Generalized Paradigm

- Sequential codes follow our generalized paradigm
  - Support global and local alignment algorithms
  - Support different gap systems: constant, linear, affine

$$
T_{i,j} = max \begin{cases} 0 \\ max_{0 \le l < j}(T_{i,l} + \theta_{i,l} + \sum_{k=l+1}^{j} \beta_{i,k}) \\ max_{0 \le l < i}(T_{l,j} + \theta'_{l,j} + \sum_{k=l+1}^{j} \beta_{k,j}) \\ T_{i-1,j-1} + \gamma_{i,j} \end{cases}
$$

Example serial code follows the paradigm

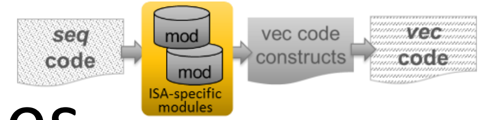
```
for i ←0; i < n+1; i++ do
  T_{0,i} = U_{0,i} = L_{0,i} = 0;
for j ←0; j < m+1; j++ do
  T_{j,0} = U_{j,0} = L_{j,0} = 0;
for i ←1; i < n+1; i++ do
  for j ←1; j < m+1; j++ do
    L_{i,j} = max(L_{i-1,j} + GAP_EXT, T_{i-1,j} + GAP_OPEN);
    U_{i,j} = max(U_{i,j-1} + GAP_EXT, T_{i,j-1} + GAP_OPEN);
    D_{i,j} = T_{i-1,j-1} + BLOSUM62_{ctoi(Q_{j-1}),ctoi(S_{i-1})};
    T_{i,j} = max(0, L_{i,j}, U_{i,j}, D_{i,j});
// resultant score is the maximum value in T
```



Vector Operation Modules

- Used to express required primitive vector operations
- Basic vector operations (e.g., load/store/add/max)
- Application-specific vector operations

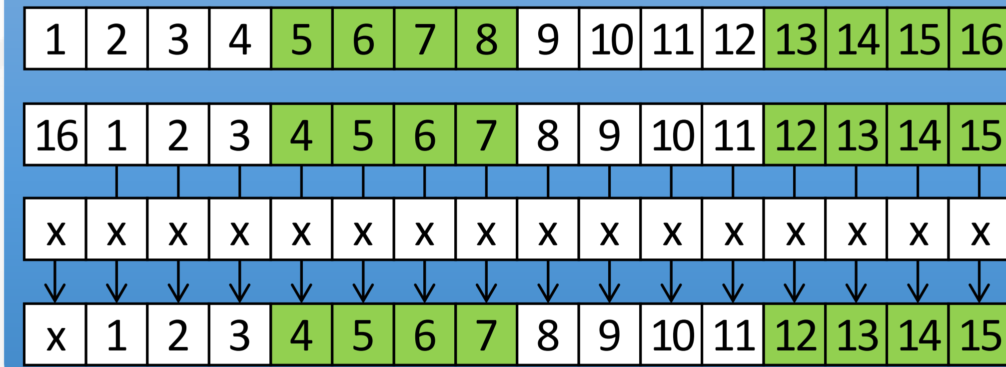
Module Name	Description
set_vector	Initialize a new vector using the given gap info.
rshift_x_fill	Right shift the vector and fill the gaps with specified value **x**
influence_test	Check if vector can affect another vector
wgt_max_scan	“weighted” max-scan over the values in a given array using vectorized method



Portability of Vector Modules

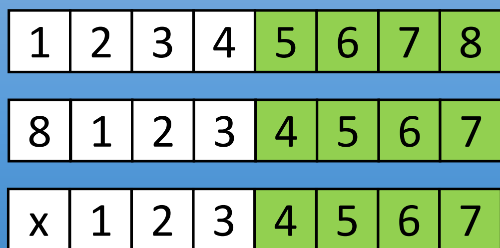
- **Example:** *rshift_x_fill()*

rshift_x_fill (IMCI 32-bit int)



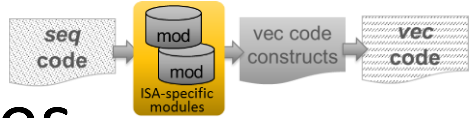
- __m512_permutevar_epi32
- __m512_set1_epi32
- __m512_mask_swizzle_epi32

rshift_x_fill (AVX2 32-bit int)



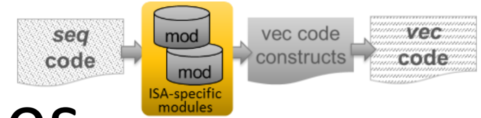
- __m256_permutevar_epi32
- __m256_insert_epi32

VirginiaTech Invent the Future
kaixihou@vt.edu
IEEE IPDPS, 5/25/2016
SyNeRGy synergy.cs.vt.edu



Portability of Vector Modules

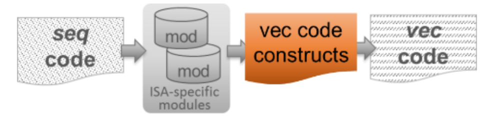
- **Example:** *rshift_x_fill()*



Portability of Vector Modules

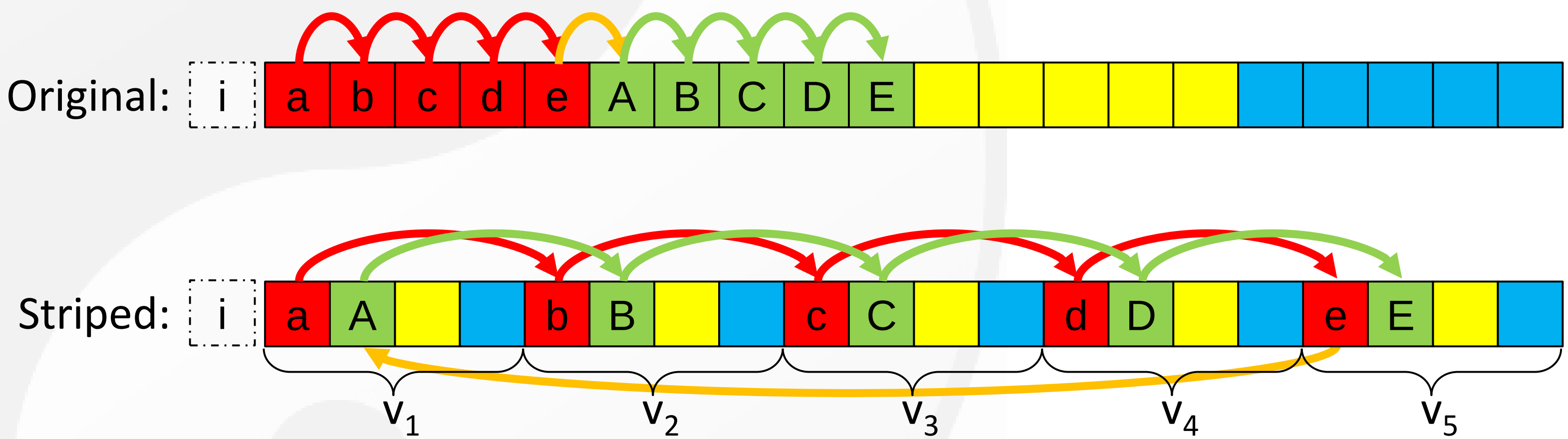
- **Example:** *rshift_x_fill()*

Provide such portability of the same functionality over different ISAs and built-in datatypes.



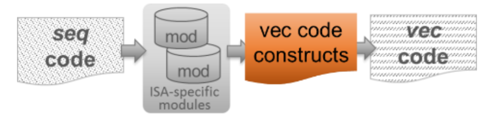
Vector Code Constructs

- Striped layout of the query sequence
  - SIMD-friendly due to the elimination of data-dependency among adjacent elements



  - Both “iterate” and “scan” methods can use the striped layout
  - Provide foundations of merging the two methods

VirginiaTech Invent the Future
kaixihou@vt.edu
IEEE IPDPS, 5/25/2016
SyNeRG synergy.cs.vt.edu



Vector Code Constructs

- Striped-Iterate
  - Iteratively correct the results if the u affect the results
- Striped-Scan
  - Correct the results using the vectorized "weighted" scan

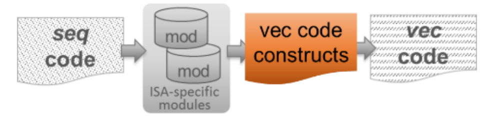
Module Name	Description
set_vector	Initialize a new vector using the given gap info.
rshift_x_fill	Right shift the vector and fill the gaps with specified value **x**
influence_test	Check if vector can affect another vector
wgt_max_scan	"weighted" max-scan over the values in a given array using vectorized method

Function aalign_iterate()
  // Preprocess the column values

```
REC_UP = rshift_x_fill(REC_UP, 1, REC_
int j = 0;
vT = load_vector(arrT2 + j * veclen);
while influence_test(REC_UP, REC_CRT) d
    vT = max_vector(vT, REC_UP);
    store_vector(arrT2 + j * veclen, vT);
    vTmax = max_vector(vTmax, vT);
    REC_UP = add_vector(REC_UP, REC_
    if ++j >= k then
        REC_UP = rshift_x_fill(REC_UP
        j=0;
    vT = load_vector(arrT2 + j * veclen);
swap(arrT1, arrT2);
```

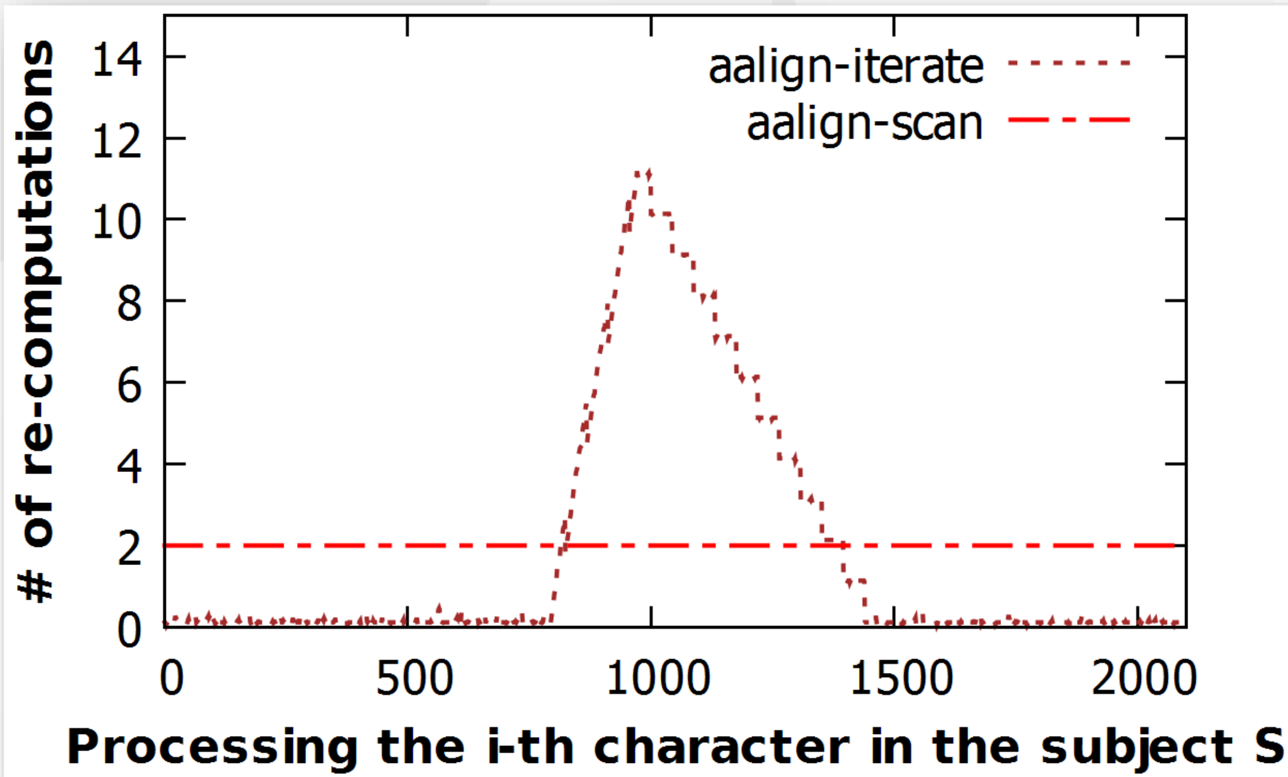
Function aalign_scan()
  // Preprocess the column values

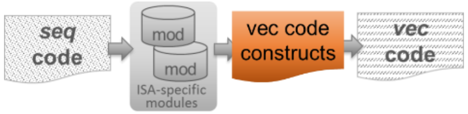
```
wgt_max_scan(arrT2, arrScan, m, INIT_T, GAP_UP_EXT, GAP_UP);

for j ←0; j < k; j++ do
    vTup = load_vector(arrScan + j * veclen);
    vT = load_vector(arrT2 + j * veclen);
    vT = max_vector(vT, vTup);
    vTmax = max_vector(vTmax, vT);
    store_vector(arrT2 + j * veclen, vT);
swap(arrT1, arrT2);
```



Hybrid Method

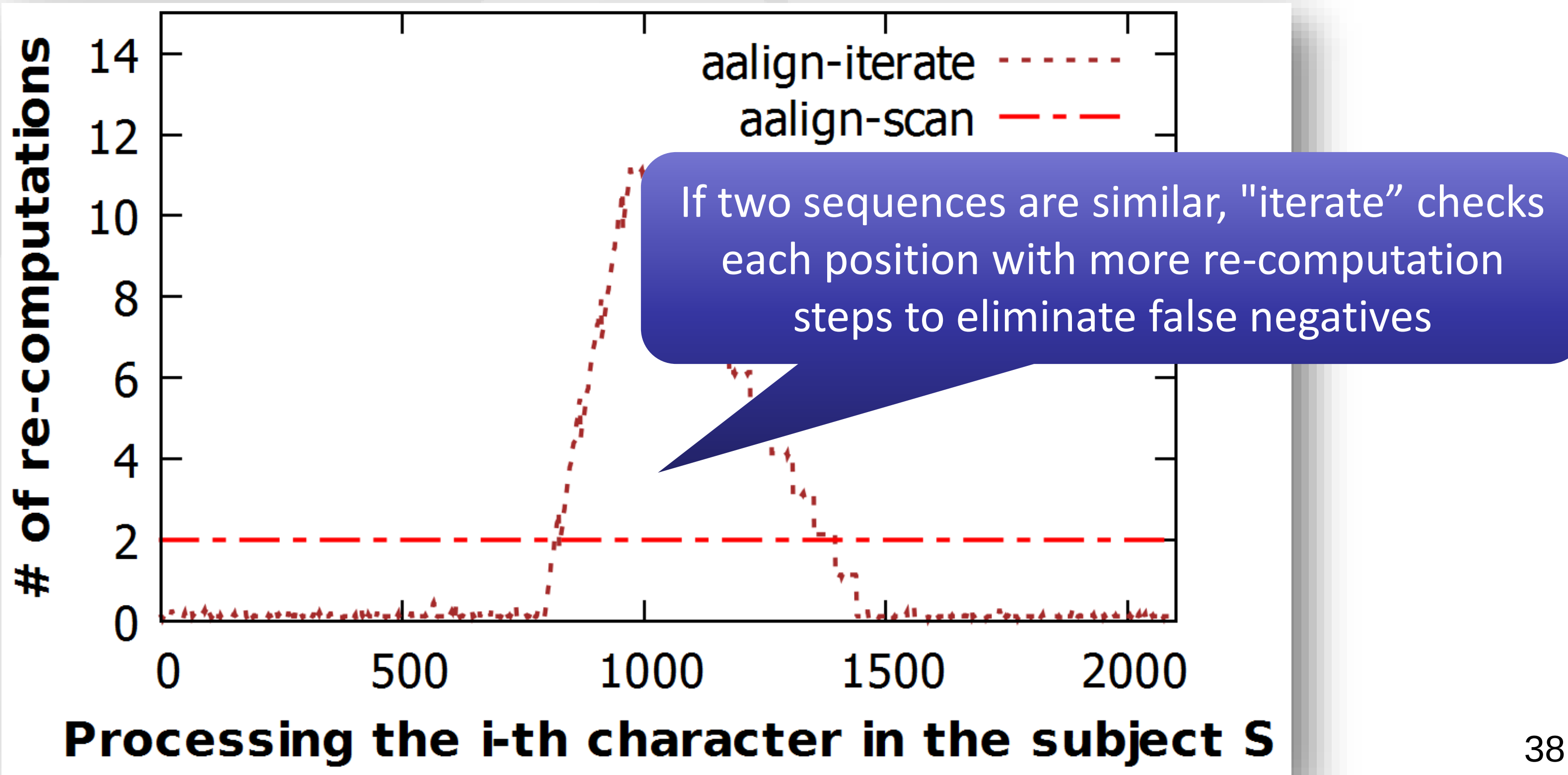
- Can we design an even better method?
- Observations (e.g., SW with affine gap)



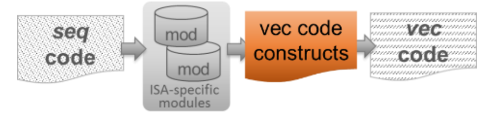


Hybrid Method

- Can we design an even better method?
- Observations (e.g., SW with affine gap)

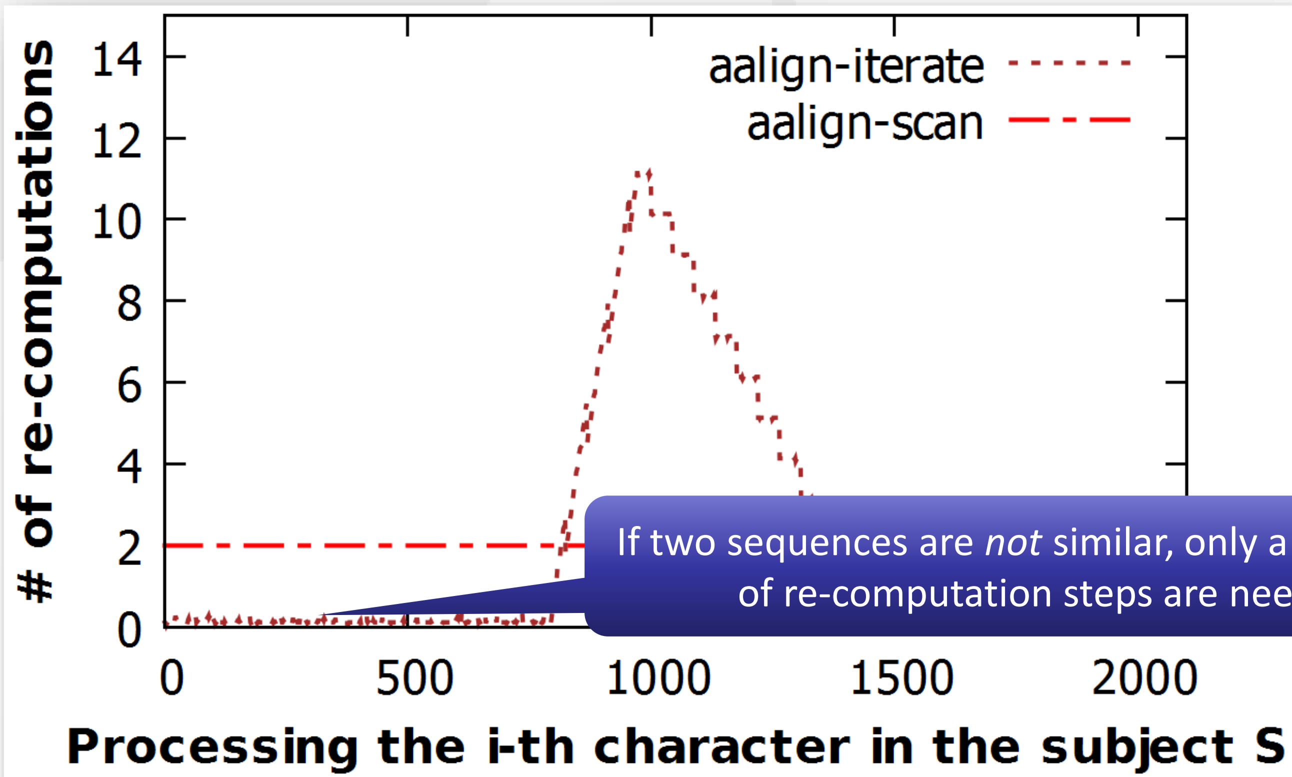


VirginiaTech Invent the Future
kaixihou@vt.edu
IEEE IPDPS, 5/25/2016
SyNeRG synergy.cs.vt.edu



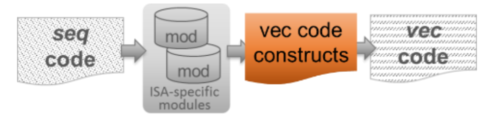
Hybrid Method

- Can we design an even better method?
- Observations (e.g., SW with affine gap)



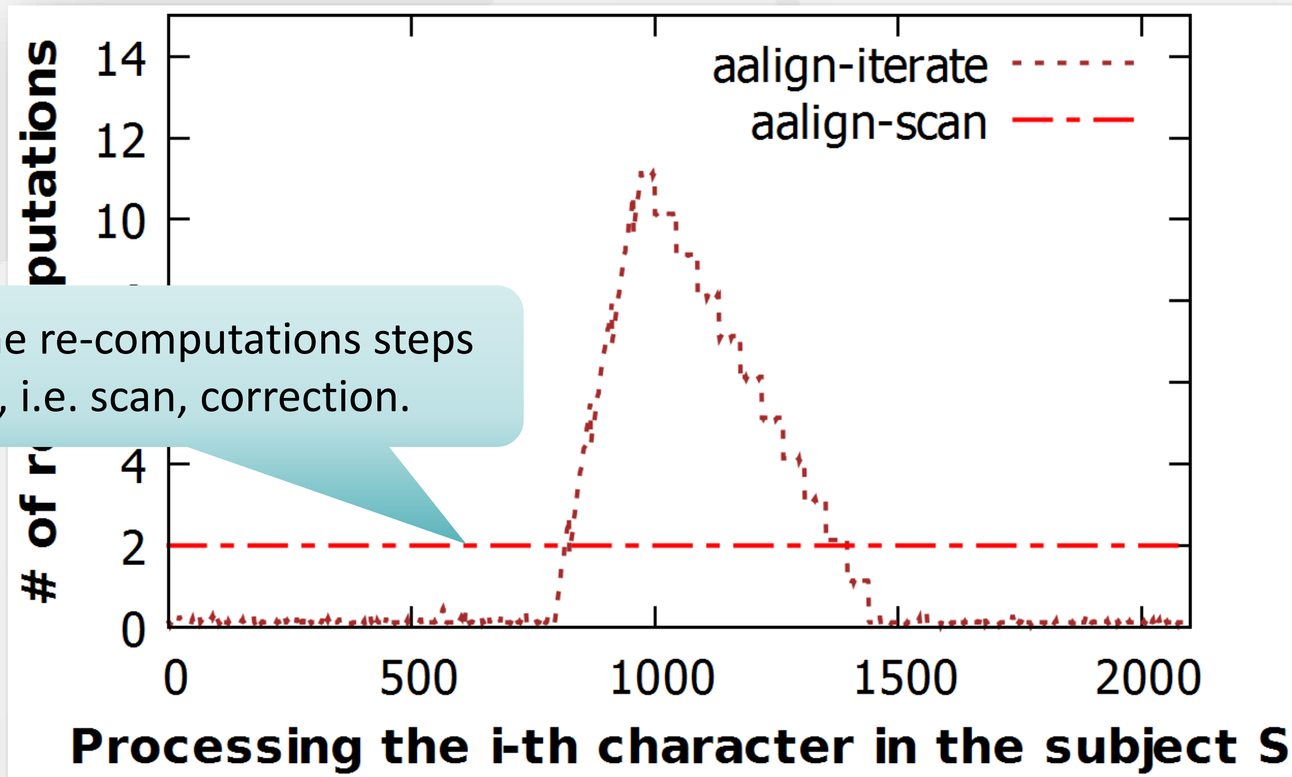
If two sequences are *not* similar, only a few rounds of re-computation steps are needed.

VirginiaTech
Invent the Future
kaixihou@vt.edu
IEEE IPDPS, 5/25/2016
SyNeRG
synergy.cs.vt.edu



Hybrid Method

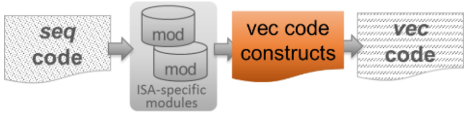
- Can we design an even better method?
- Observations (e.g., SW with affine gap)



In "scan", the re-computations steps are fixed, i.e. scan, correction.

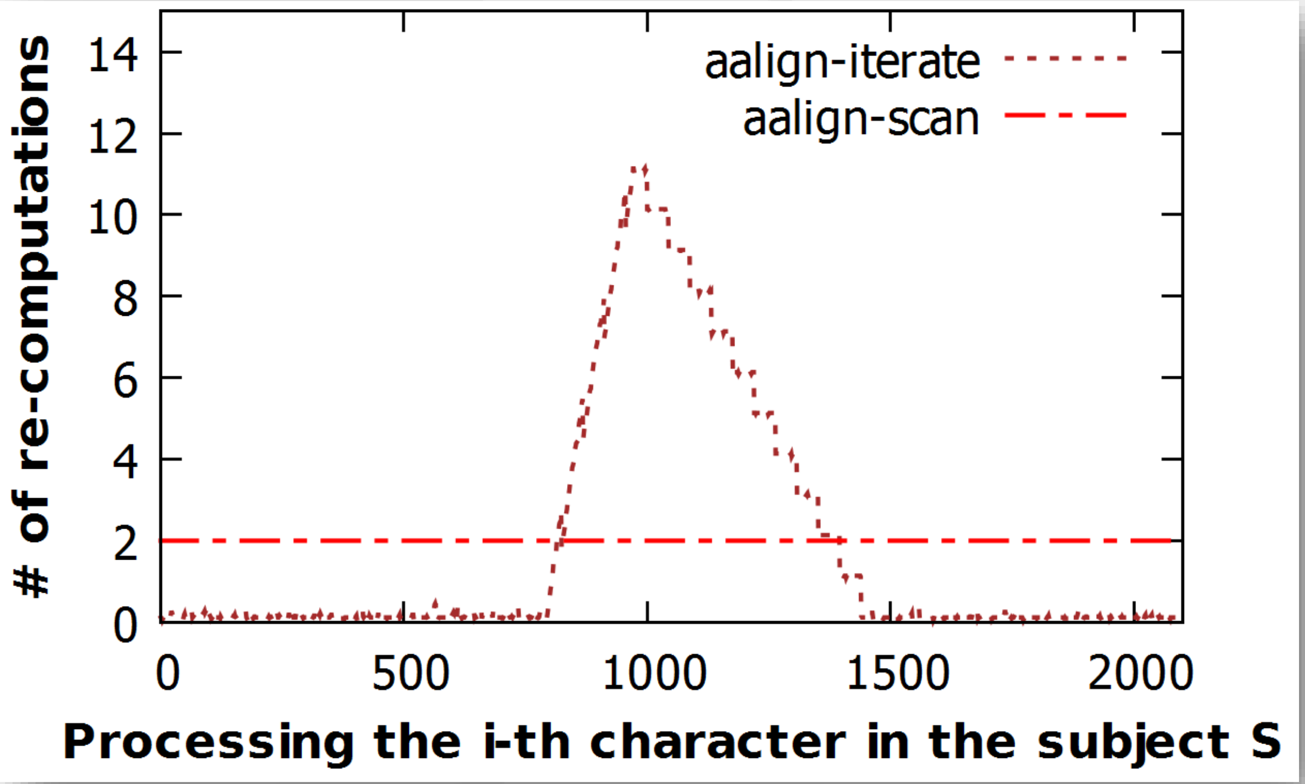
VirginiaTech
Invent the Future
kaixihou@vt.edu
SyNeRG
synergy.cs.vt.edu

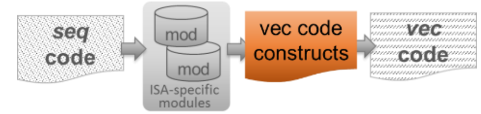
IEEE IPDPS, 5/25/2016



Hybrid Method

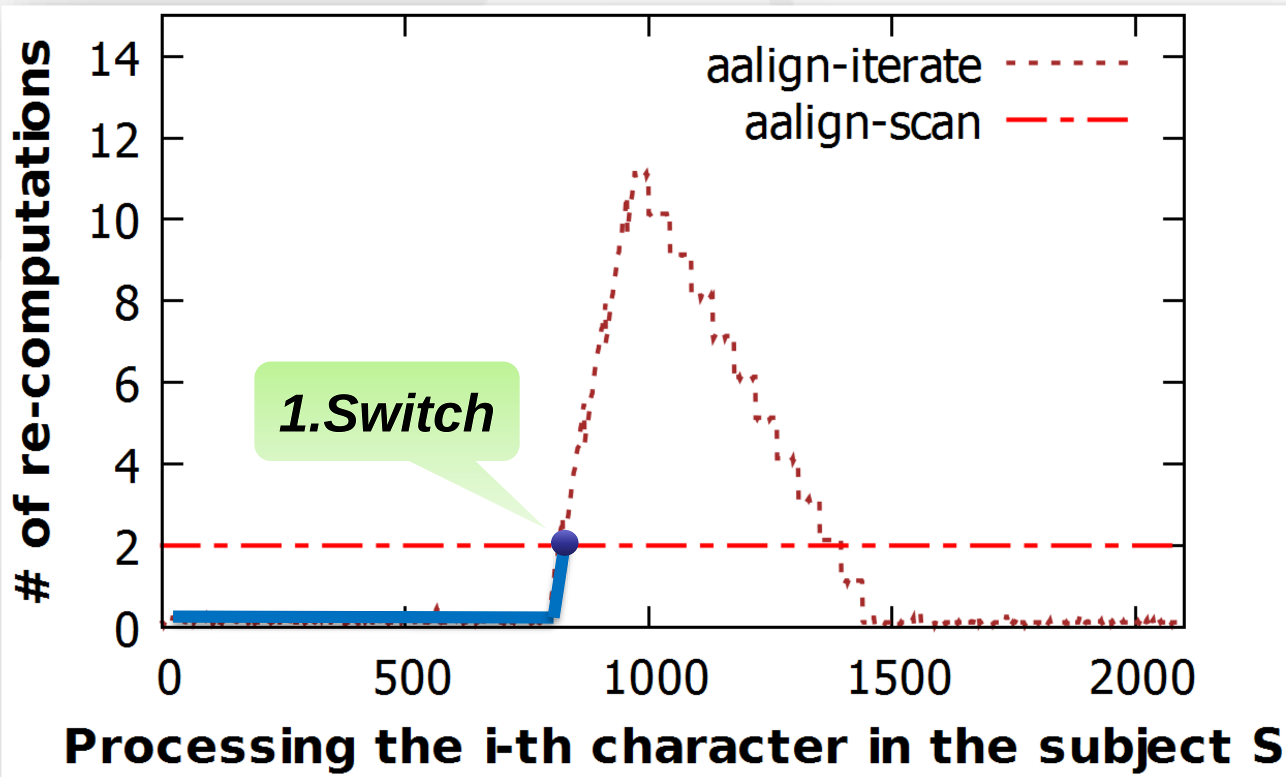
- Our idea: Automatically switch to the better strategy based on the current # of re-computations at runtime





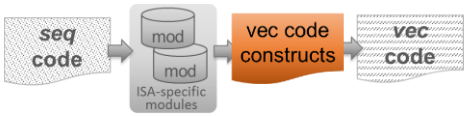
Hybrid Method

- Our idea: Automatically switch to the better strategy based on the current # of re-computations at runtime

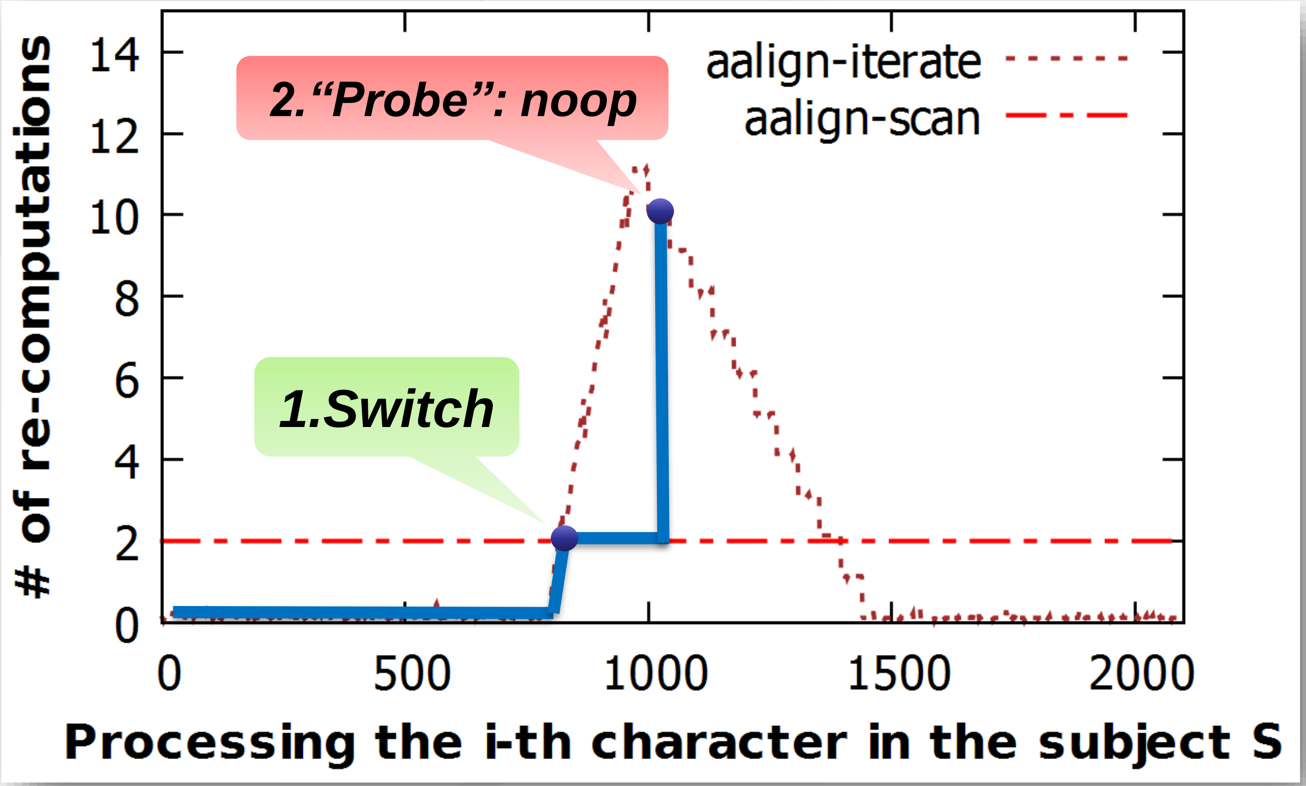


kaixihou@vt.edu
IEEE IPDPS, 5/25/2016

Hybrid Method



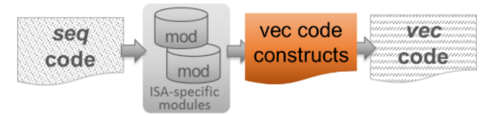
- Our idea: Automatically switch to the better strategy based on the current # of re-computations at runtime



kaixihou@vt.edu
IEEE IPDPS, 5/25/2016

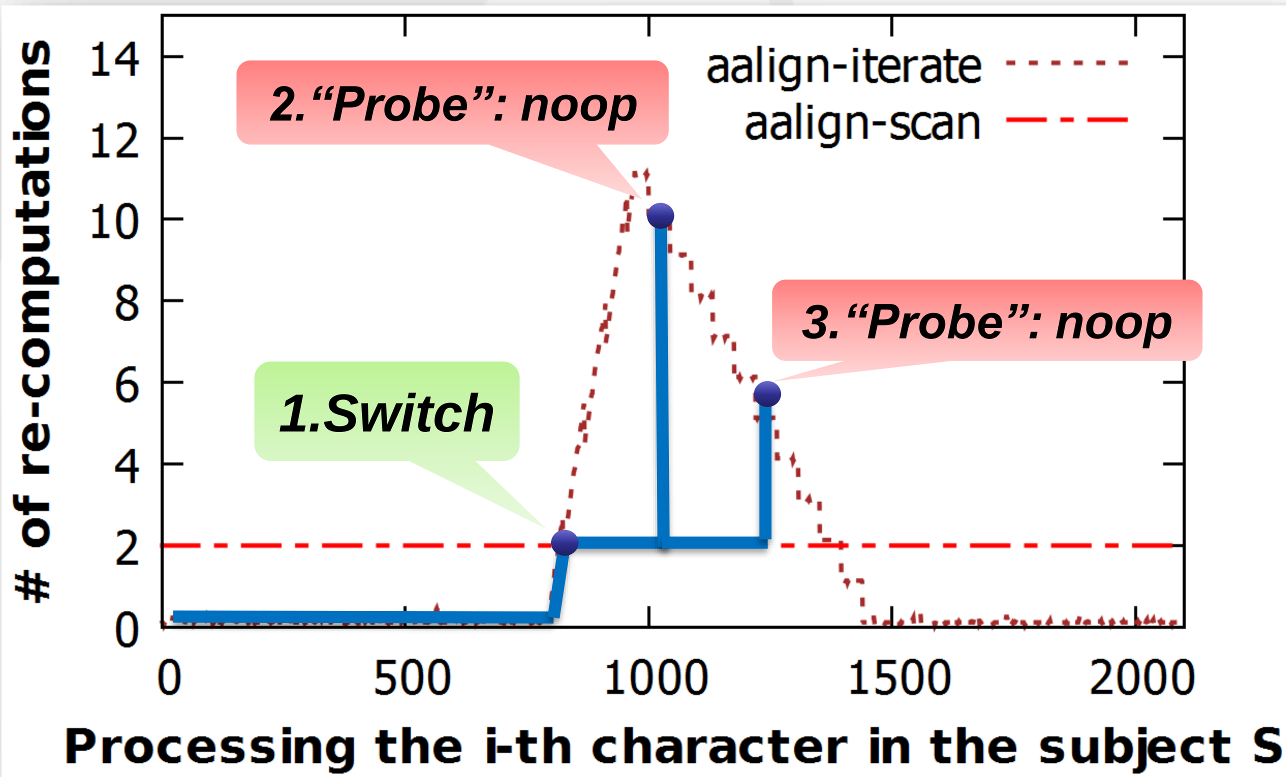
VirginiaTech Invent the Future

SyNeRG synergy.cs.vt.edu

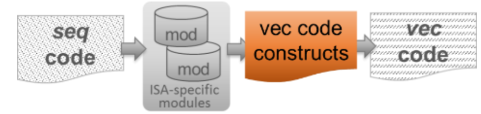


Hybrid Method

- Our idea: Automatically switch to the better strategy based on the current # of re-computations at runtime

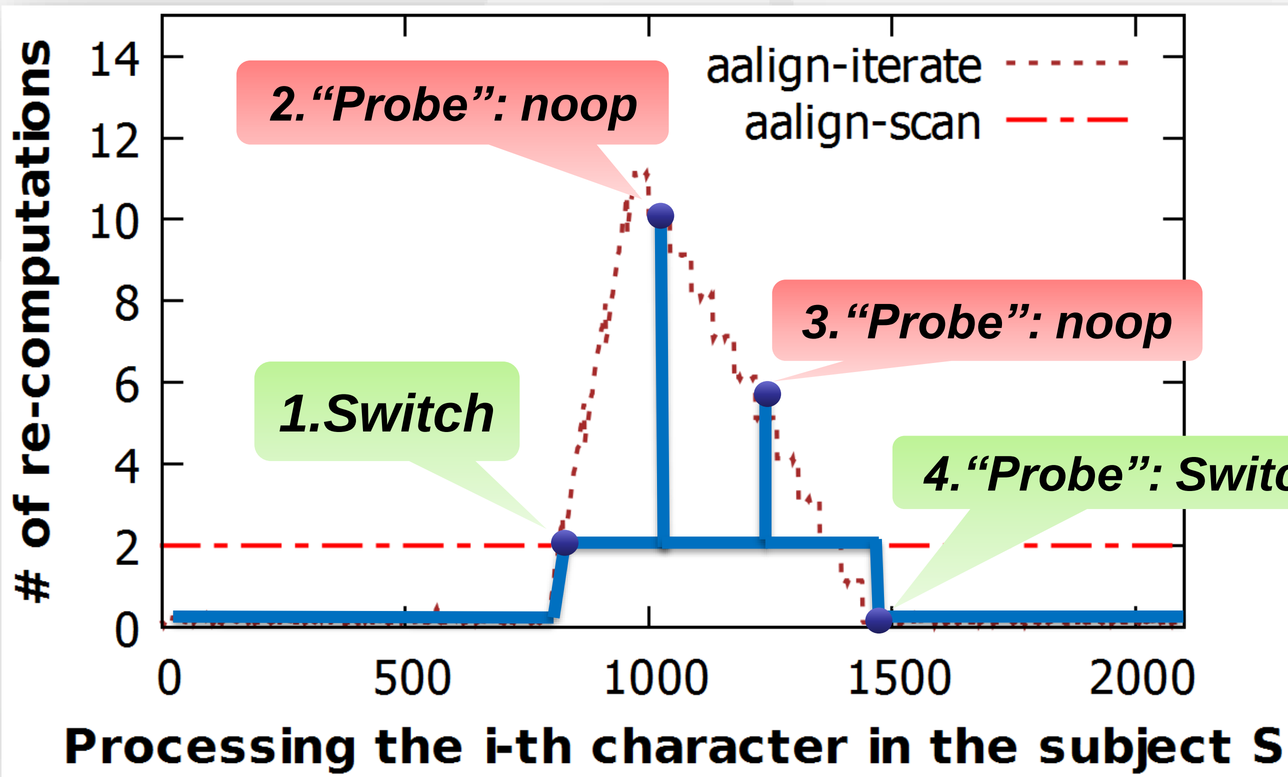


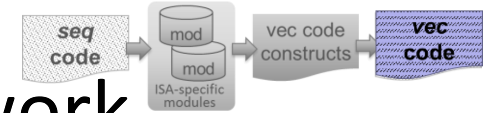
kaixihou@vt.edu



Hybrid Method

- Our idea: Automatically switch to the better strategy based on the current # of re-computations at runtime



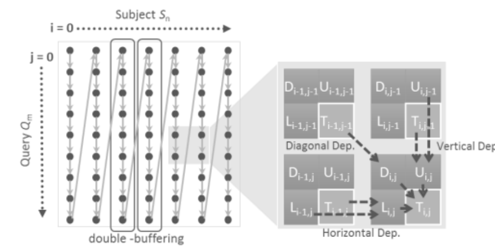


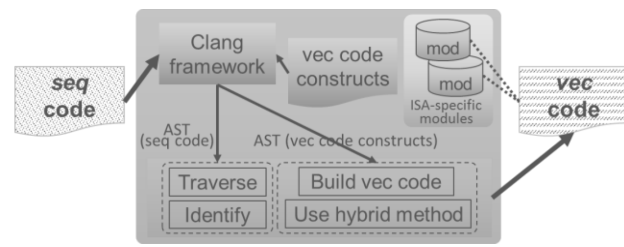
Details of the AAlign Framework

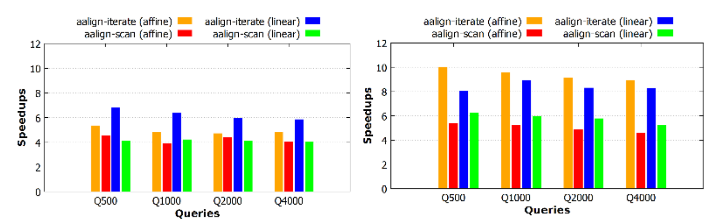
- Code translation
  - Use clang driver to create Abstract Syntax Tree (AST)
  - Detect the type of sequential code (e.g., SW or NW; linear or affine gap; etc.)
  - Create real vector codes based on vector code constructs and type information
- Multi-threaded version
  - Perform one-to-all sequence alignment
  - Database sequences have been sorted for better load balance

Roadmap

- Introduction & Motivation
- Background
  - Vector ISA
- AAlign Framework
  - Generalized Paradigm
  - Vector Modules
  - Vector Code Constructs
  - Hybrid Method
- **Evaluation & Discussion**
- Conclusion







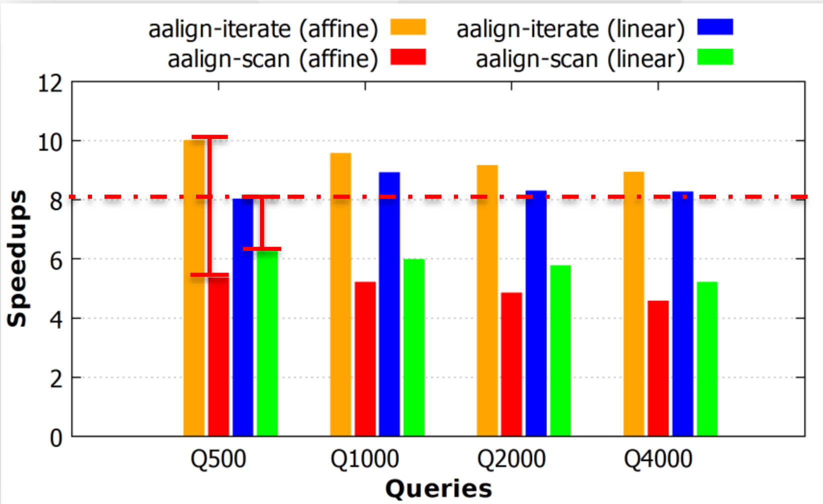
Evaluation & Discussion

- Experiment Setup
  - The queries are real sequences selected from NCBI, i.e., AL4A1_HUMAN(Q500), COSA1_HUMAN(Q1000), B0I1R8_HUMAN(Q2000), MUC17_HUMAN(Q4000)
  - The database is “Swiss-prot” containing over 570k sequences

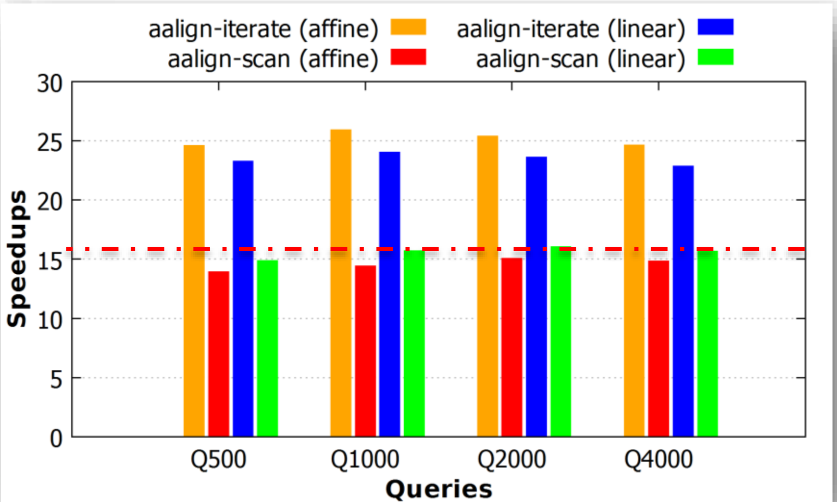
Parameter	CPU	MIC
Product Name	Intel Xeon E5-2680 v3	Intel Xeon Phi 5110P
Code Name	Haswell	Knights Corner
# of Cores	24	60
Clock Rate	2.5 GHz	1.05 GHz
L1/L2/L3 Cache	32 KB/ 256 KB/ 30 MB	32 KB/ 512 KB/ -
Memory	128 GB DDR3	8 GB GDDR5
Compiler	icpc 15.3	icpc 15.3
Compiler Options	-xCORE-AVX2 –O3	-mmic -O3
Vector ISA	AVX2	IMCI

Evaluation & Discussion

- Speedups over Serial Counterparts



SW on Haswell CPU

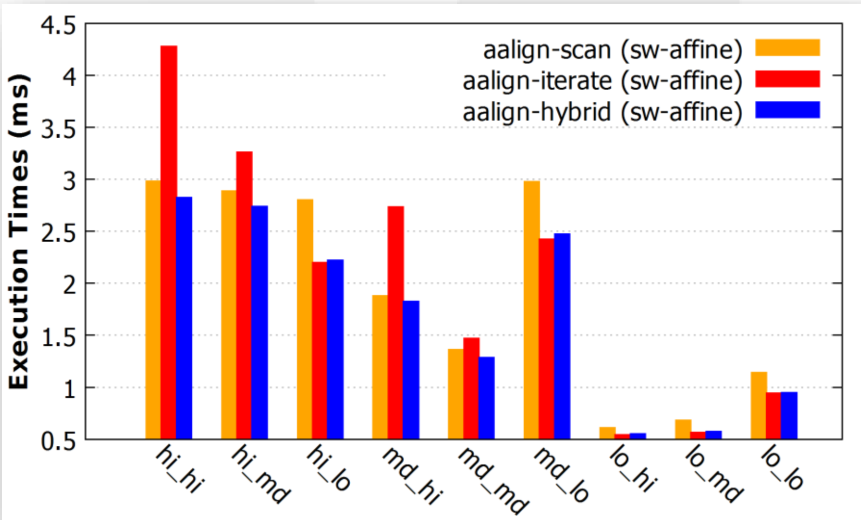


SW on Knights Corner MIC

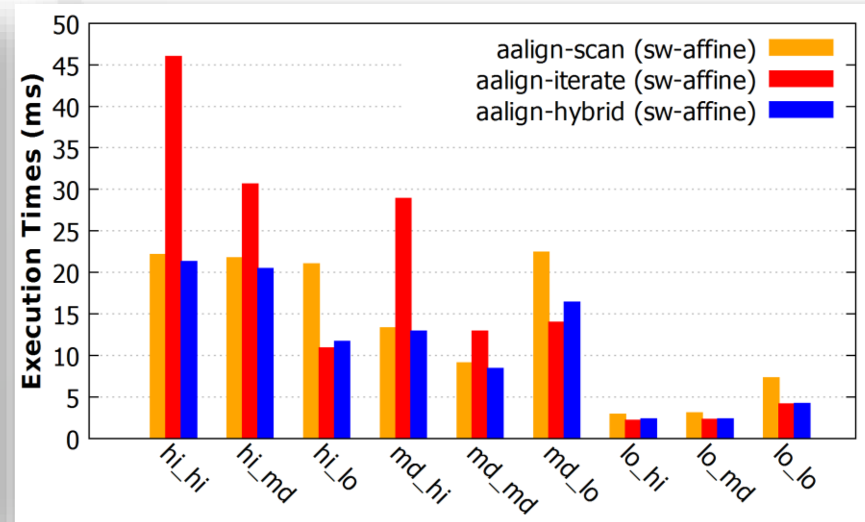
  - "AAlign-Iterate" can achieve better vectorization efficiency
  - Superlinear speedups for "AAlign-Iterate" comes from the elimination of a considerable amount of re-computation when the *influence_test()* fails

Evaluation & Discussion

- Performance of Hybrid Method



SW on Haswell CPU

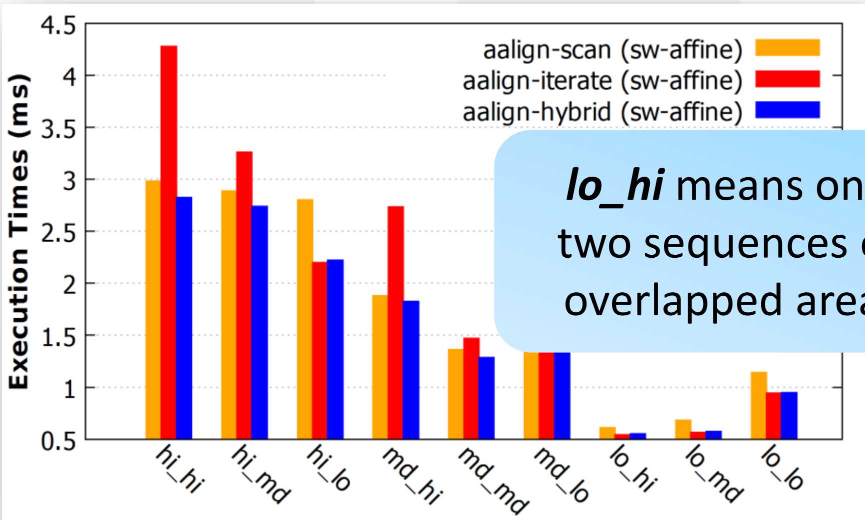


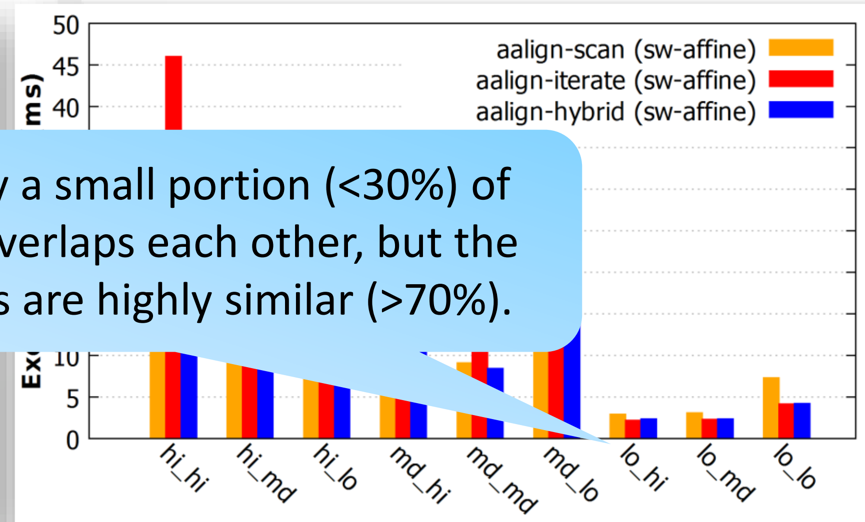
SW on Knights Corner MIC

 – Use query coverage (**QC**) and max identity (**MI**) to describe the similarity of two sequences. (Format: <QC>_<MI>, e.g., lo_hi)

Evaluation & Discussion

- Performance of Hybrid Method





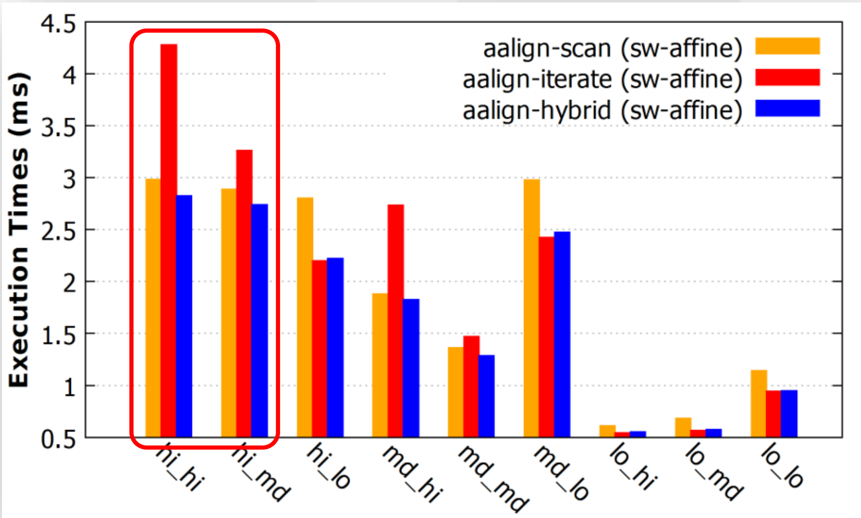
lo_hi means only a small portion (<30%) of two sequences overlaps each other, but the overlapped areas are highly similar (>70%).

SW on Haswell CPU ***SW on Knights Corner MIC***

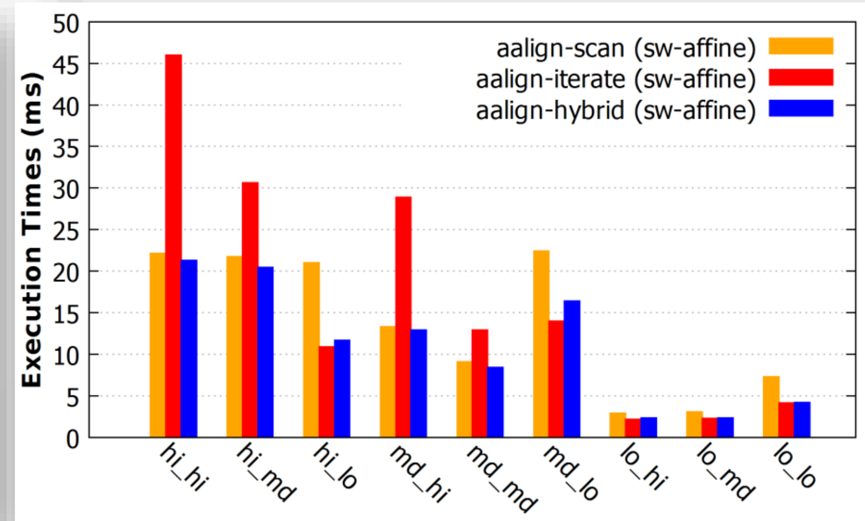
  - Use query coverage (**QC**) and max identity (**MI**) to describe the similarity of two sequences. (Format: <QC>_<MI>, e.g., lo_hi)

Evaluation & Discussion

- Performance of Hybrid Method



SW on Haswell CPU

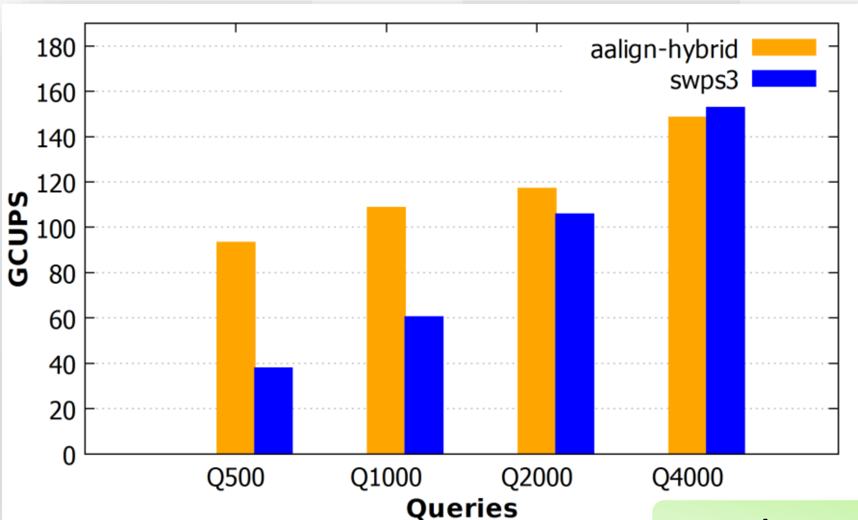


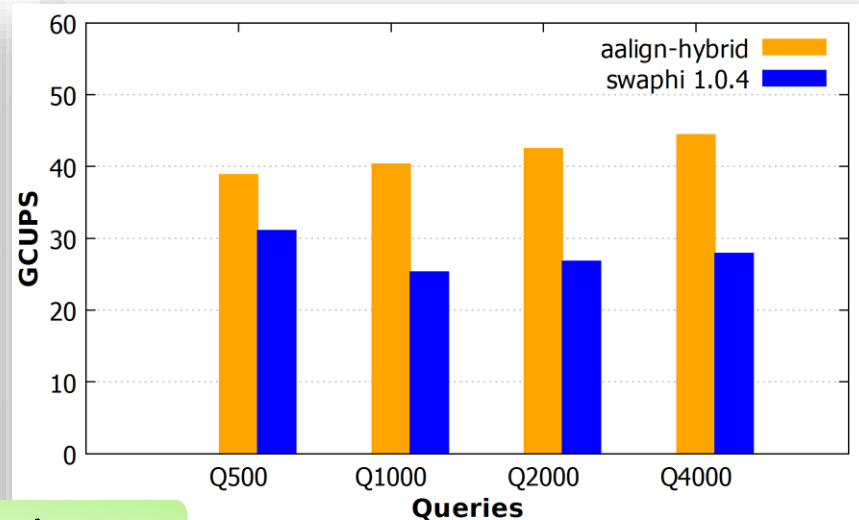
SW on Knights Corner MIC

- Use query coverage (**QC**) and max identity (**MI**) to describe the similarity of two sequences. (Format: <QC>_<MI>, e.g., lo_hi)
- Hybrid method can achieve better performance than both vector algorithms; for some cases, it can approximate the superior one

Performance Comparison with Open-Source Tools

- AAlign vs. SWPS3* on CPU   AAlign vs. SWAPHI** on MIC





Higher the better

SW on Haswell CPU   ***SW on Knights Corner MIC***

  - CPU: AAlign codes can outperform SWPS3 by up to 2.5x
  - MIC: AAlign codes can outperform SWAPHI by up to 1.6x

*A. Szalkowski, C. Ledergerber, P. Krhenbhl, and C. Dessimoz, "SWPS3 fast multi-threaded vectorized Smith-Waterman for IBM Cell/B.E. and 86/SSE2," BMC Res Notes, 2008

**Y. Liu and B. Schmidt, "SWAPHI: Smith-waterman protein database search on Xeon Phi coprocessors," Int'l Conf. on Application-specific Systems, Architectures and Processors (ASAP), 2014.

Conclusion

- AAlign: A specialized framework for pairwise alignment algorithms on the x86-based processors
  - Efficient vector codes based on "striped-iterate" & "striped-scan"
  - Sets of platform-specific vector modules
- Design: A new input-agnostic hybrid method
- Performance:
  - Significant performance gains over serial counterparts
  - Auto-switching to better vectorization strategy at runtime (hybrid method)
  - Up to 2.5x performance benefit over existing multi-threaded tools
- Availability: https://github.com/vtsynergy/aalign

THANK YOU! More info: http://synergy.cs.vt.edu