Georgia Tech Institute for Data Engineering and Science

- Data Science Foundations
- Data Driven Discovery

HIGH PERFORMANCE COMPUTING FOR BIOLOGY AND MEDICINE

SRINIVAS ALURU

GEORGIA INSTITUTE OF TECHNOLOGY

CREATING THE NEXT®

Broad Scope

- Application Domains
  - Omics: Genomics, Transcriptomics, Metagenomics, Microbiomics, systems Biology, Metabolomics, Proteomics
  - Digital diagnosis, Drug-receptor interactions, Medical imaging, Edge computing, Life in simulation
- Middleware
  - Optimized libraries, New language?
- Architecture
  - Processor, accelerator
  - Compute in memory
  - Memory, Storage (SSD, NVM), Network Interconnect, Cloud
- Security, Privacy

Broad Scope

- Application Domains
  - Omics: Genomics, Transcriptomics, Metagenomics, Microbiomics, ~~systems Biology, Metabolomics, Proteomics~~
  - ~~Digital doctor, Drug-receptor interactions, Medical imaging, Edge computing, Life in simulation~~
- Middleware
  - Optimized libraries, New language?
- Architecture
  - Processor, accelerator
  - Compute in memory
  - ~~Memory, Storage (SSD, NVM), Network Interconnect, Cloud~~
- Security, Privacy

Next Generation Sequencing – The Big Data Challenge

Then (2005)

ABI 3700

96 ~800 bp reads

76.8 X 10^3 bases

~$1 per kilo base

No

Illumina NovaSeq 6000

20 billion 2X150-bp paired reads

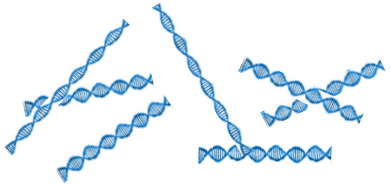
3000 X 10^9 bases

~$1 per 100 million bases

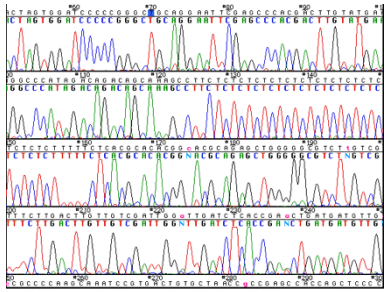
Overview of NGS Bioinformatics

DNA (e.g. Chromosomes)

Fragmentation into small DNA segments



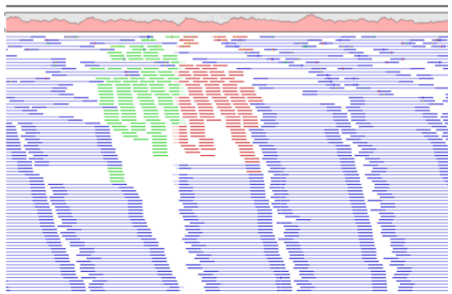
High-throughput Sequencing



Base Calling & Error Correction

TAGTGGATCCCCGGGC
TGCAG...
GGCCCATAGACAGACA
GCAAA...
TCTCTCTTTTTCTCAC
GCACA...
TTTTCTTGACTTGTT
TCGAT...

Mapping reads to



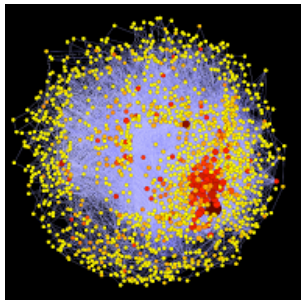
Overview of NGS Bioinformatics

Reads

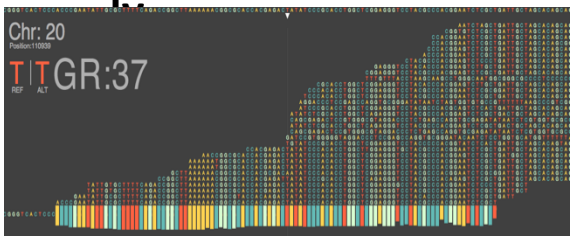
TAGTGGSATCCCCGGGC
TGCAG...
GGCCCATAGACAGACA
GCAAA...
TCTCTCTTTTTCTCAC
GCACA...
TTTTCTTGACTTGTTG
TCGAT...

Transcriptome Assembly

Digital Gene Expression



Genome Assembly

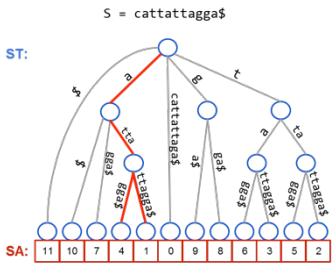


Gene Networks

Metagenomic Clustering & Assembly

Compression

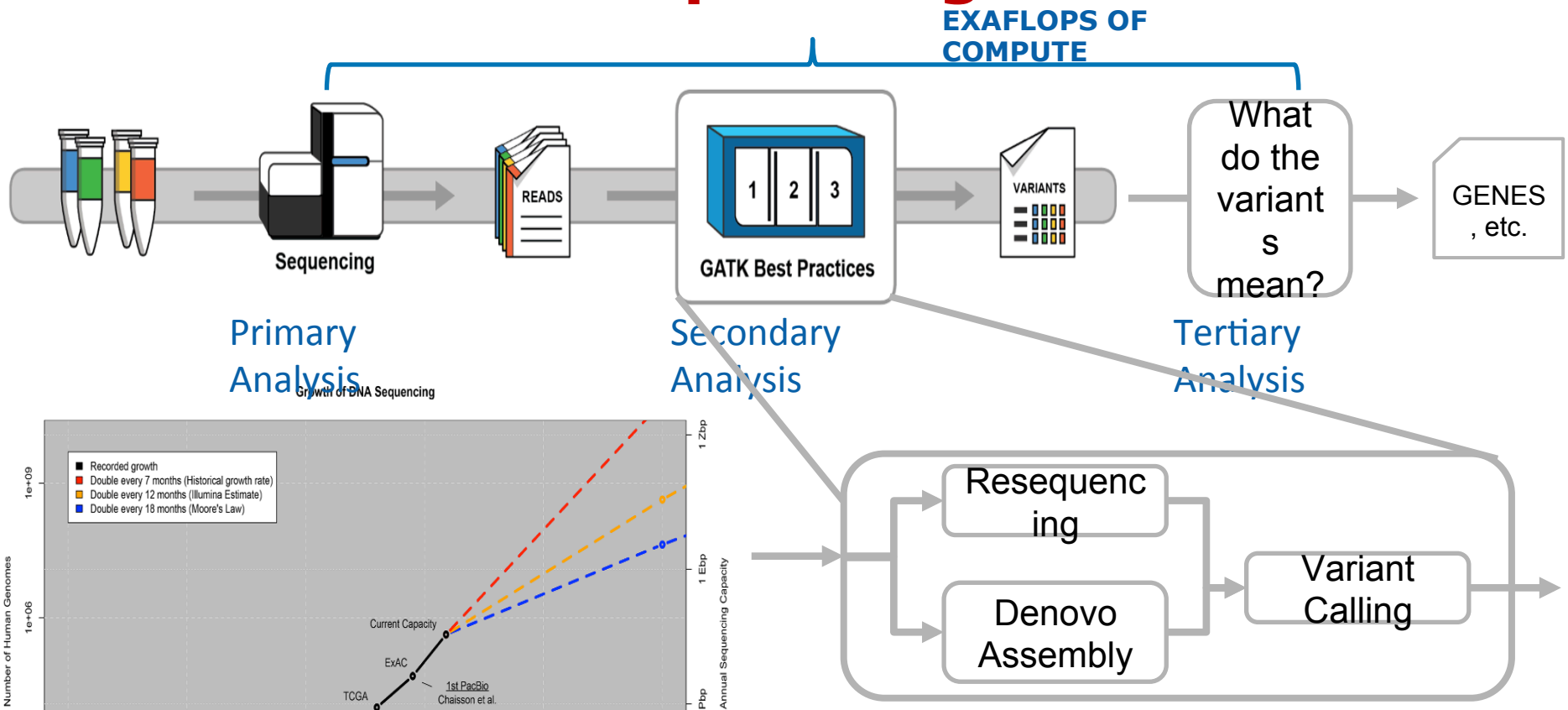
Indexing



Pattern Matching

Genome Analysis

Genomics, Metagenomics, Microbiomics, Transcriptomics = Next Generation Sequencing



Stephens, et. al. Big Data: Astronomical or Genomical?. PLOS Biology. (2015)

Dozens of Tools But Computation Limited to a Few Building Blocks

	Applications	Tools	Building Blocks: Smith Waterman	FM-index	Kmer counting and indexing	Hidden Markov Model	Sorting	Machine (Deep) Learning
Traditional Genomics	Sequence Alignment	Exact sequence alignment	100%					
		HMMER				100%		
NGS Secondary Analysis			20-50%	40-80%				
				100 %				
			5-55%		30-90%			
					65-90%			
	Calling		20-30%		2-10%	35-60%		
		Deepvariant						100%
NGS Tertiary Analysis		e.g. CANDLE						100%

Sequence Mapping	80.5-98.2 %
Denovo Assembly	64-99.4%
Variant Calling	72-93%

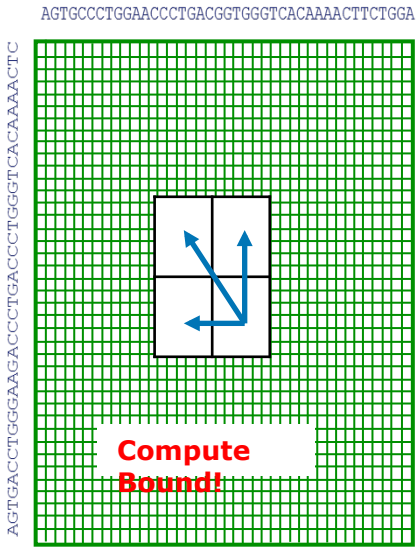
…putational Buildin… …ough … GS Secondary Ana… …ls.

Md Vasimuddin, Sanchit Misra, Srinivas Aluru. bioRxiv 301903

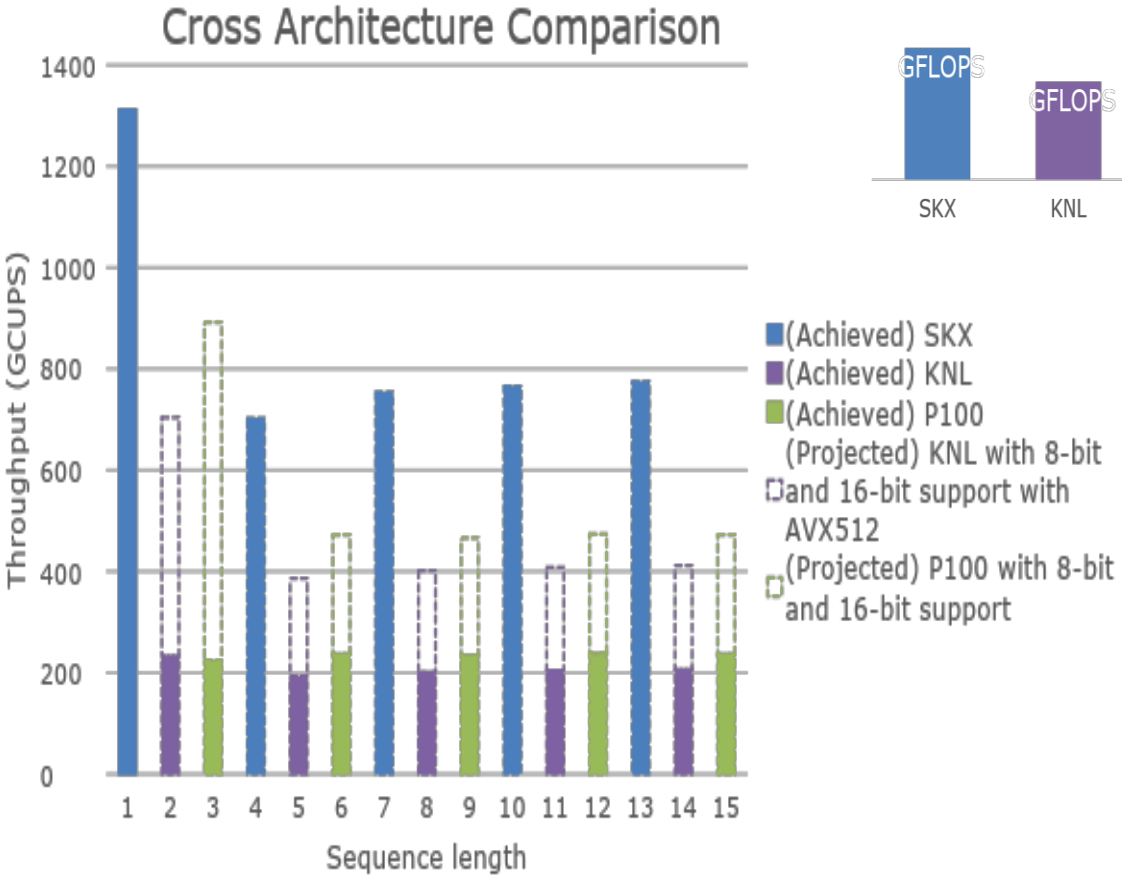
- *To identify the* performance characteristics and architectural bottlenecks of the key building blocks, *we performed a cross architecture comparison* using their optimized implementations
- *In the process, we also identified the most efficient of the mainstream*

Smith Waterman – Many Flavors

Basic Smith Waterman – no backtrack



- Only need to store one row at a time
- The row fits in cache
- Parallelize (vectorize and multithread) across multiple matrices
- Only challenge: variability in matrix sizes – sorting helps
- 12 operations (6 int max, 4 int add, 1 cmpeq, 1 blend) per cell update

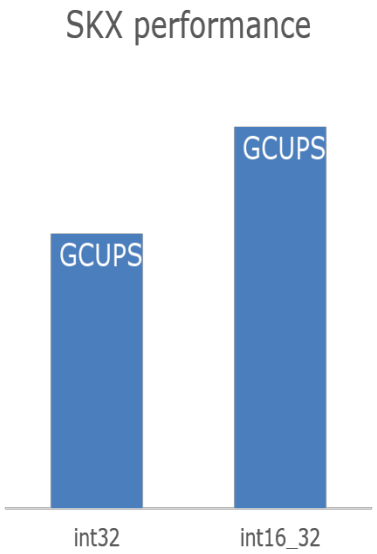


- *Dataset: Synthetic sequences obtained by sampling sequences of various lengths from human genome hg38 chromosome 1*
- *SKX: Intel® Xeon® Platinum 8180 Processor (Skylake), KNL: Intel® Xeon Phi™ Processor 7250 (Knights Landing), P100: Nvidia Pascal P100 GPU*
- *SKX uses AVX512 runs int8 implementation for length 100 and int16 for all other lengths*
- *KNL runs int8 with AVX2 for length 100 and int32 with AVX512 for all other lengths*
- *P100 results obtained by running Smith-Waterman benchmark from NVBIO and uses int32 operations.*

Smith Waterman – Many Flavors

SW in HaplotypeCaller

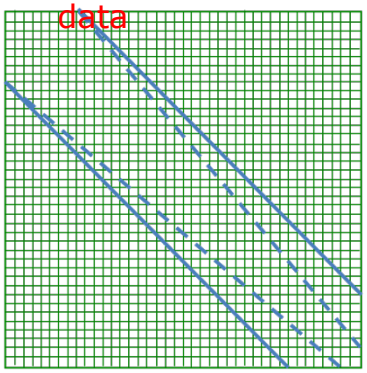
- Needs backtracking information
- Unlike the basic SW where best score is the max of all cells, best score is the max of the cells of the last row and columns
- Maximum score values need > 16 bits (~17-19 bits)
- 27 operations (5 cmpgt, 7 add, 2 and, 5 blend, 5 max, 2 or, 1 andnot)
- Compute bound
- We achieved 157.4 GCUPS at 84% efficiency compared to peak achievable.



Dataset: Real data intercepted from HaplotypeCaller run. Machine: SKX

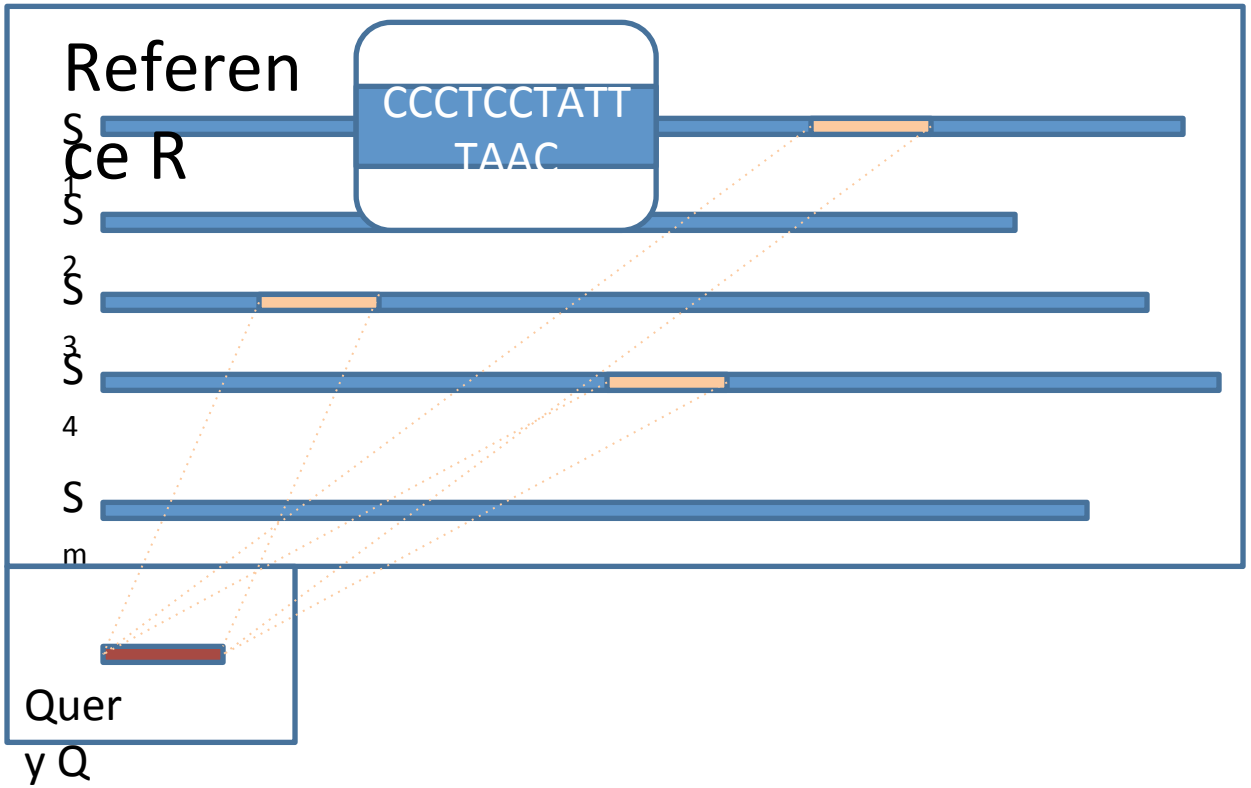
SW in BWA-MEM

- No backtracking needed
- 8-bit precision is sufficient
- Only a diagonal band is computed
- Size of the band changes from top to bottom
- Various conditions for early exit if satisfactory alignment is not possible
- Parallelism within one matrix is limited
- Vectorization across matrices is hampered by irregularity
- Could be compute or BW bound depending on data



- Dataset: Real data intercepted from BWA-MEM run. Machine: SKX.
- We have achieved only 5 GCUPS so far by vectorization across matrices.
- Benefit of vectorization is only 10x despite vector width of 64.
- Given the limited vectorization opportunity, this is a

FM Index Based Sequence Search – many Flavors

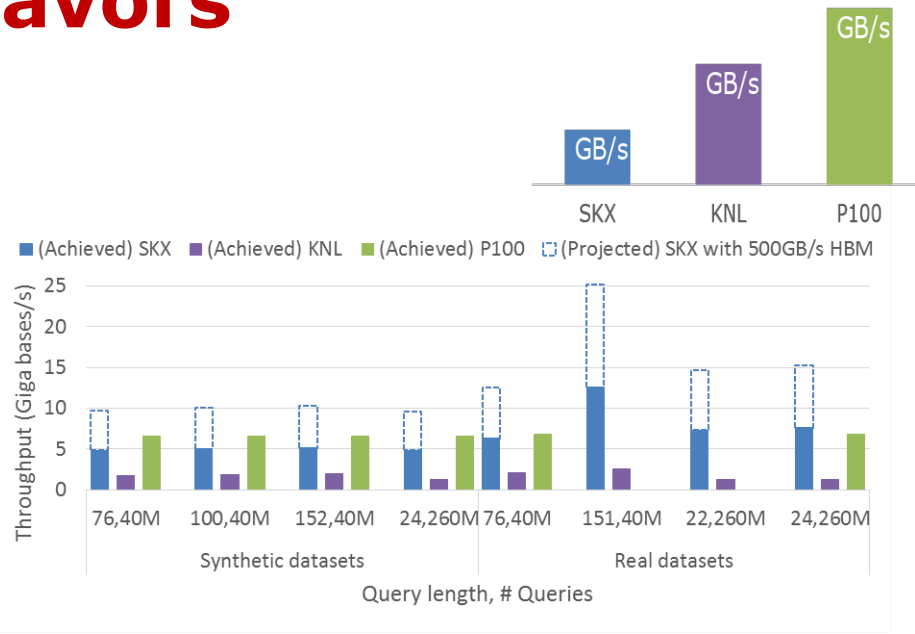


- |Q| < 200, |R| is in Billions; e.g.; human genome is 3 Billions in length.
- Exact search: Finding exact matches of end-to-end Q in R
- Inexact search: Allow a few mismatches or insertions/deletions while still requiring end-to-end match
- SMEM search: At each base of Q, find the matches in R for the longest substring of Q passing through that base that has matches in R

FM Index Based Sequence Search – Many Flavors

Exact search

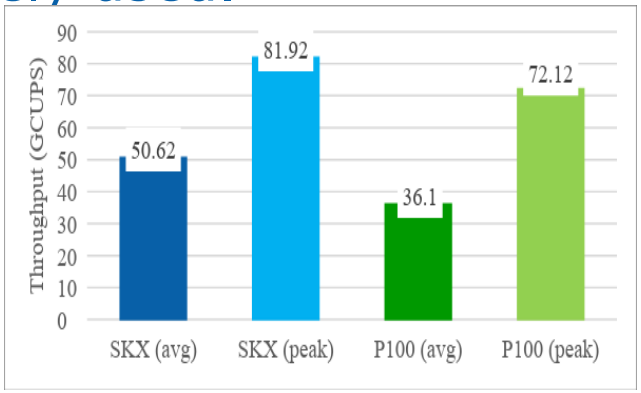
- Reads two memory locations in a data structure of size several GBs for each base of the query
- Performs very little compute
- The computation for one base calculates the memory locations to be read for the next base
- There is no locality between subsequent memory accesses resulting in a lot of data coming from memory
- Due to data dependency, vectorization within the processing of one query is not possible.
- Vectorization across queries is beneficial by irregular access to memory



- SKX and KNL implementations need extensive software prefetching, while P100 hides memory latencies through scheduling other threads. In either case, the problem is bandwidth bound!
- SKX performs comparable to P100, despite 1/3 the BW, due to larger caches thus needing lower BW.
- KNL-like HBM on SKX with peak BW of 500 GB/s can increase SKX performance by 2x.
- While for each memory access, 64B (full cache line) are read, we only need 4B. So, a compute in memory that can combine multiple such requests and puts only the required data on the bus has potential to increase throughput by up to 16x

Pairwise Hidden Markov Model (PairHMM)

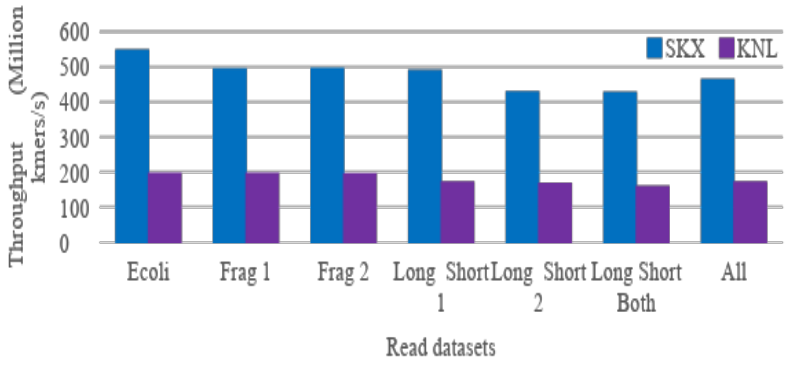
- Very similar to basic SW.
  - Also a DP based algorithm that populates a matrix for a pair of sequences.
- Only main difference is that computation is done on FP numbers.
  - Only multiply and add operations are used
  - Computation is mostly SP
  - DP is rarely used.

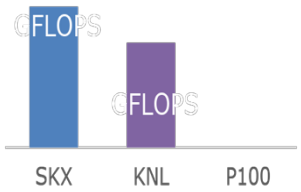


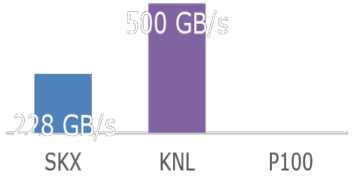
Dataset: Real data obtained by intercepting PairHMM stage in

Kmer Counting

- Performed using hash table as underlying data structure
- Hash function computation is performed using vectorization
  - BW bound on SKX as the amount of computation performed per byte of data read is lower than the compute to BW ratio of SKX
  - Compute bound on KNL as HBM on KNL provides much higher BW
- Insertion into hash table is performed using radix sort based hashing scheme
  - The hash table is multiple GBs in size and access to it is completely irregular.
  - There is very little compute making the problem bandwidth bound. Only 16B out of 64B (cache line) accessed are needed.
  - Vectorization across multiple insertions to hash table is hampered by irregular memory access.
  - Extensive software prefetching needs to be used







- To the best of our knowledge, no performant stand alone implementation for kmer counting exists for GPGPUs
- SKX performs better despite the lower memory BW due to larger caches hence requiring less BW

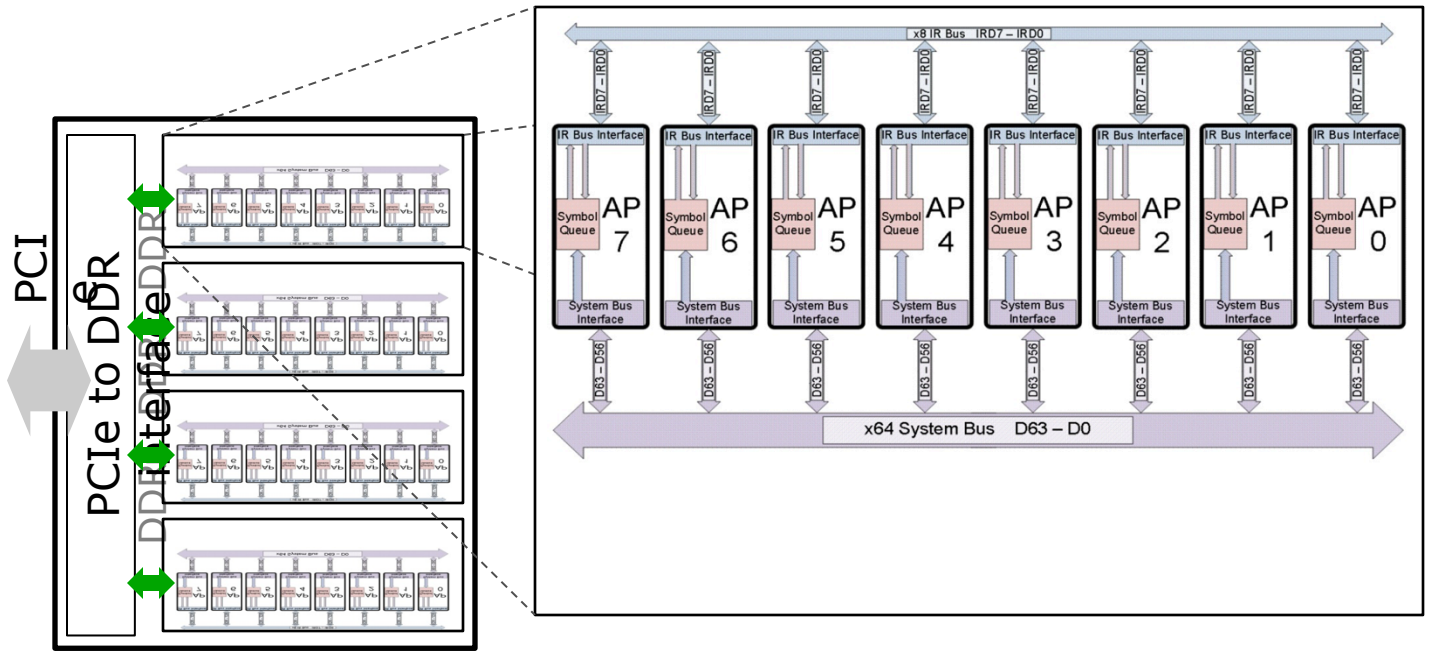
Performance Characteristics of NGS Secondary Analysis

- Irregular compute and irregular memory access to large data structures is quite rampant
- Thus available opportunity of vectorization is very limited
  - However, modern general purpose processors rely on vectorization to extract performance
- Irregular memory access to large data structures requires extensive use of software prefetching on SKX and KNL. P100 hides the latency by scheduling other threads
  - For either architecture, such problems are BW bound
  - The BW bound problem is further amplified by the fact that only a part of the cache line that is accessed is used
- Integer operations at 8-bit, 16-bit and 32-bit level are quite useful
  - max, add, cmp, and, or, xor, permute, blend, shift, etc. are particularly useful
  - Since DNA alphabet can be represented using 2-bits, support for 2-bit level cmp operations will also help
- FP operations are used less often
  - PairHMM is the only kernel that uses FP operations and the operations are limited to only multiply and add
  - None of the expensive numerical floating point operations like division, square

Model Architecture for NGS Secondary Analysis?

- Architecture Model:
  - Data movement is a big cost (both power and time) and the architecture should minimize the amount of data movement and the associated cost
  - The architecture should not rely on vectorization for improving performance or reducing power requirement
  - The architecture should avoid requiring software prefetching as it adds to the cost in terms of number of instructions
  - More fine grain control of cache content would be helpful – e.g. scratchpad memory instead of cache
- Memory
  - HBM needed!
  - Compute in memory: that can combine data from multiple requests for smaller data sizes before putting them on the bus thus optimizing BW usage
- Precision levels and Operations
  - Integers:
    - 2-bit, 4-bit (?), 8-bit, 16-bit, 32-bit, some support for 64-bit operations.
    - max, add, cmp, and, or, xor, permute, blend, shift.
  - Floating point: SP multiply and add, very little support for DP multiple

Micron/NIS Automata Processor



Automata Processor board

Rank of Automata Processor chips

Capabilities:

Capacity	**1,572,864 million STEs**
Compilation time	**Automata dependent**
Load time	**50 milliseconds**
Data Processing	**1 Gbps to 8 Gbps**

Protein motif Search

Protein Automata

A database of known protein motifs

For example, tachykinins are known to excite neurons and contract smooth muscles. Tachykinins have been shown to modulate pain, reduce blood pressure, and increase sperm motility.

Sequence analysis has detected many members of the tachykinin family:

Substance P from mammals, birds and fish.
Neurokinin A from mammals, birds and fish.
Neurokinin B from mammals and frogs.
Kassinin from frogs.
Hylambatin from frogs.
Phyllomedusin from a frog.
Physalaemin from a frog.
Ranamargarin from a Chinese frog.
Uperolein from frogs.
Ranatachykinins A to D from frogs.
Scyliorhinins from dogfish.
Carassin from goldfish.
Eledoisin from octopus.

African rhacophorid frog

hum

Georgia Tech College of Computing
Computational Science and Engineering

Conversion of a protein motif to automaton

Protein Automata

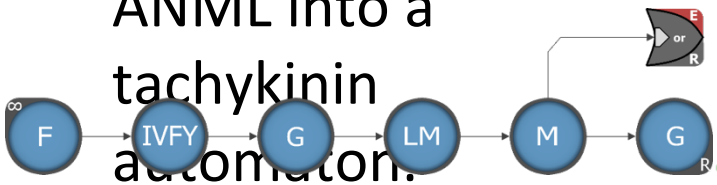
1. tachykinin motif in ProSite.

F-[IVFY]-G-[LM]-M-[G>].

2. PROTOMATA converts the ProSite pattern into ANML.

```
<automata-network name="tachykinin" id="tachykinin">
  <state-transition-element id="T0" symbol-set="F"
start="all-input">
    <activate-on-match element="T1"/></state-transition-
element>
  <state-transition-element id="T1" symbol-set="[IVFY]">
    <activate-on-match element="T2"/></state-transition-
element>
  <state-transition-element id="T2" symbol-set="G">
    <activate-on-match element="T3"/></state-transition-
element>
  <state-transition-element id="T3" symbol-set="[LM]">
    <activate-on-match element="T4"/></state-transition-
element>
  <state-transition-element id="T4" symbol-set="M">
    <activate-on-match element="T5"/>
    <activate-on-match element="T6"/>
  </state-transition-element>
  <or id="T5"><report-on-high/></or>
  <state-transition-element id="T6" symbol-set="G">
    <report-on-match/>
  </state-transition-element>
</automata-network>
```

3. AP SDK turns ANML into a tachykinin automaton.

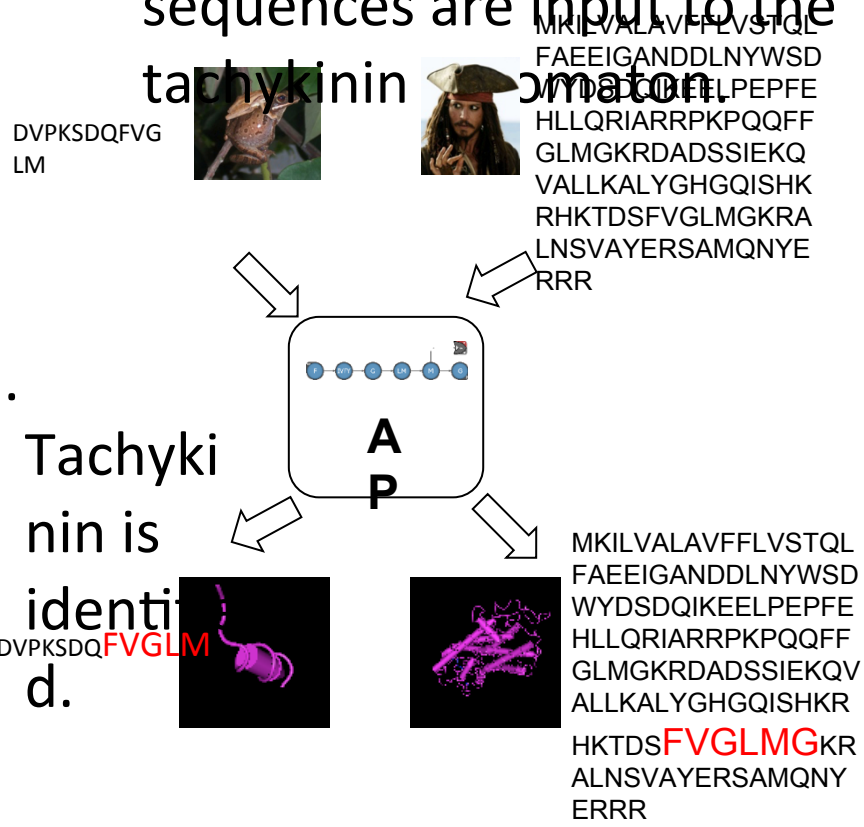


Georgia Tech College of Computing
Computational Science and Engineering

Identifying motifs in protein sequences

Protein Automata

4. Uncharacterized sequences are input to the tachykinin automaton.



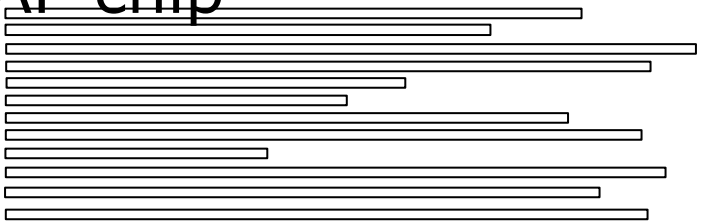
Georgia Tech College of Computing
Computational Science and Engineering

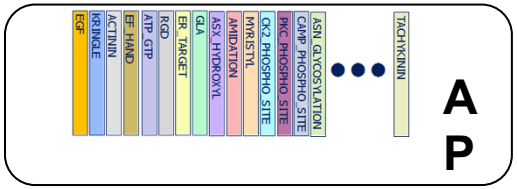
Scanning Entire Database in Parallel

Protein Automata

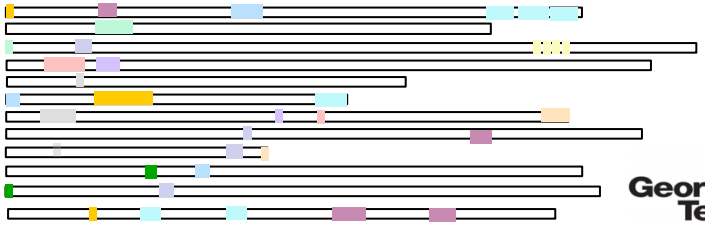
ProSite database currently has 1308 motifs
Entire database can be programmed using a single AP chip

uncharacterized protein sequences





all motifs identified



Georgia Tech College of Computing
Computational Science and Engineering

Acknowledgements

- Tony Pan
- Ankit Srivastava
- Sanchit Misra, Intel
- Indranil Roy, Natural Intelligence Semiconductor

QUESTIONS?