Georgia Tech Institute for Data Engineering and Science

GENOMES GALORE: BIG DATA CHALLENGES IN THE LIFE SCIENCES

SRINIVAS ALURU
GEORGIA INSTITUTE OF TECHNOLOGY

CREATING THE NEXT®

The Big Data Challenge

Then (2005)

ABI 3700

96 ~800 bp reads
76.8 X 10^3 bases
~$1 per kilo base

Now

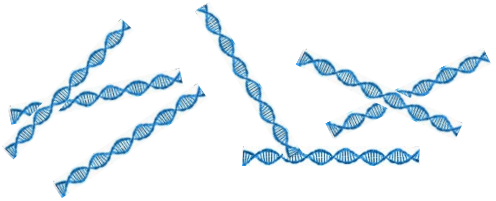
Illumina Hiseq X

6 billion 2X150-bp paired reads
1800 X 10^9 bases
~$1 per 100 million bases

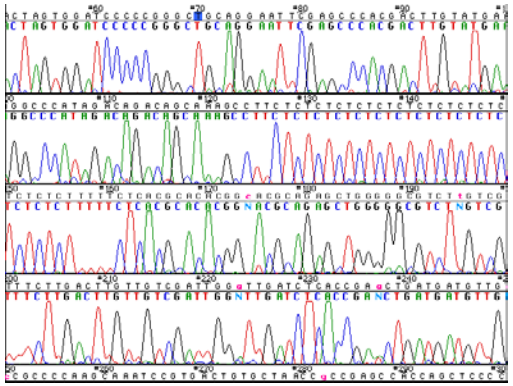
Overview of NGS Bioinformatics

DNA (e.g. Chromosomes)

Fragmentation into small DNA segments



High-throughput Sequencing



Base Calling & Error Correction

TAGTGGATCCCCGGGCTGCAG…
GGCCCATAGACAGACAGCAAA…
TCTCTCTTTTTCTCACGCACA…
TTTTCTTGACTTGTTGTCGAT…
.
.
.

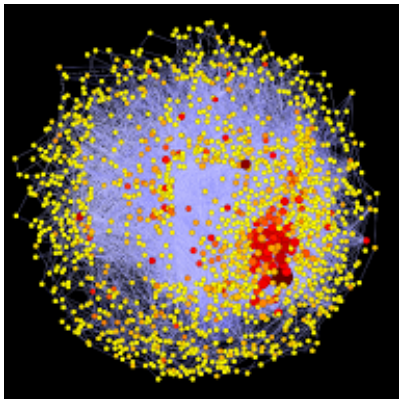
Overview of NGS Bioinformatics

Reads

TAGTGGATCCCCGGGCTGCAG…
GGCCCATAGACAGACAGCAAA…
TCTCTCTTTTTCTCACGCACA…
TTTTCTTGACTTGTTGTCGAT…
.
.
.

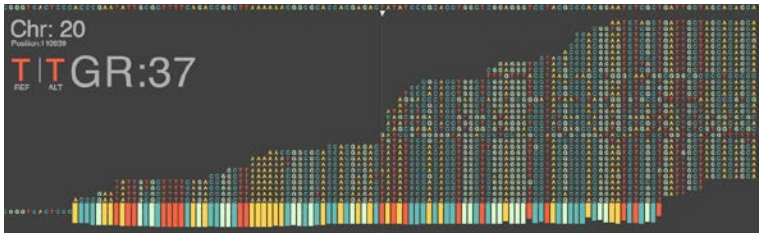
Transcriptome Assembly

Digital Gene Expression

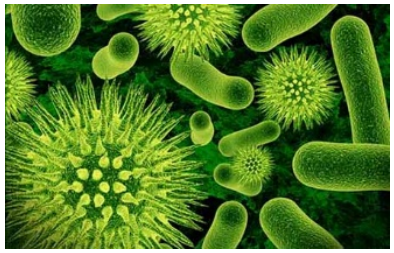


Gene Networks

Genome Assembly

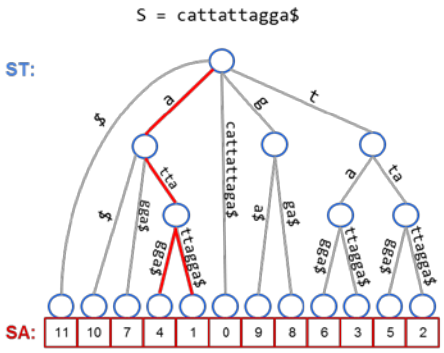


Metagenomic Clustering & Assembly



Compression

Indexing



Pattern Matching

Genome Analysis

A Variety of Sequencers

- Pacbio

  ~8,000 - 16,000 bp

- Ion Torrent/Proton

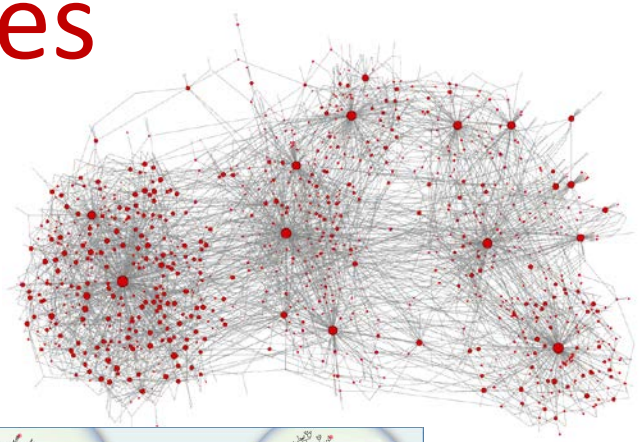
  200 bp avg

- Oxford Nanopore

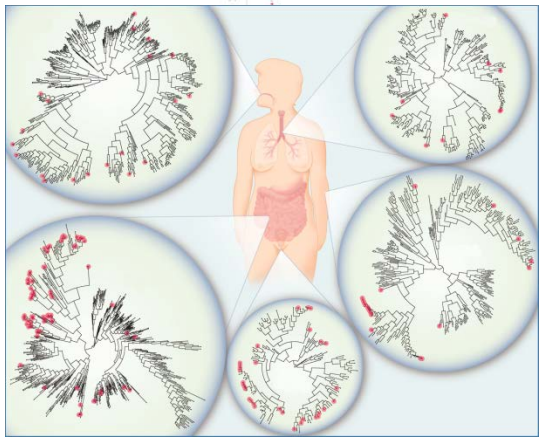
  MinION

  ~10,000 bp

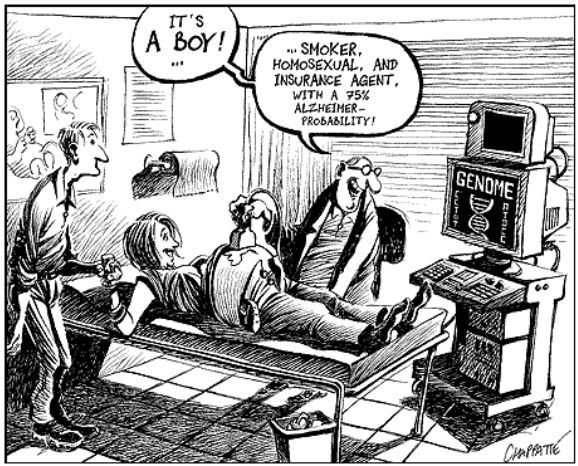
Grand Challenges



- Viral – disease transmission, outbreak detection
- Microbial – soil metagenomics, human microbiome, engineer microbes for human needs, bioterrorism



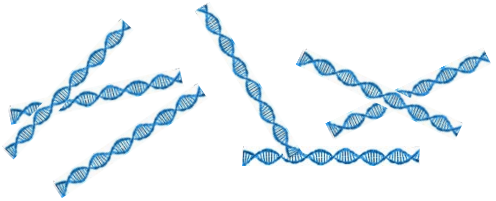
- Encyclopedia of life – sequence genomes of all organisms
- Agriculture – engineer plant genomes for optimal yield
- Precision medicine – find signatures of diseases, determine targeted cures



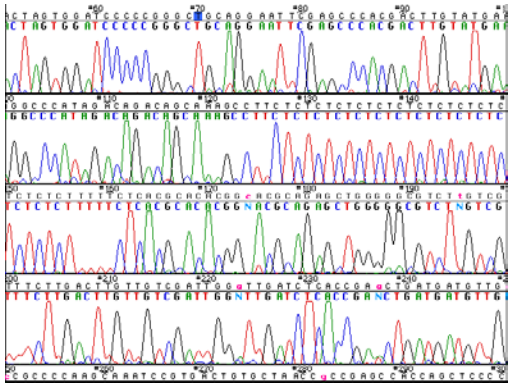
HPC Research in NGS Bioinformatics

DNA (e.g. Chromosomes)

Fragmentation into small DNA segments



High-throughput Sequencing



Base Calling & Error Correction

[IPDPS 2012
HiPC 2015]

TAGTGGATCCCCGGGCTGCAG…
GGCCCATAGACAGACAGCAAA…
TCTCTCTTTTTCTCACGCACA…
TTTTCTTGACTTGTTGTCGAT…
.
.
.

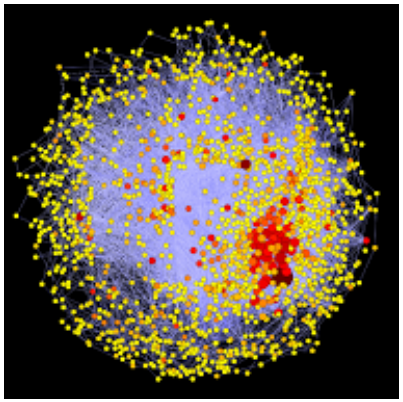
HPC Research in NGS Bioinformatics

Reads

TAGTGGATCCCCGGGCTGCAG…
GGCCCATAGACAGACAGCAAA…
TCTCTCTTTTTCTCACGCACA…
TTTTCTTGACTTGTTGTCGAT…
.
.
.

Transcriptome Assembly

Digital Gene Expression

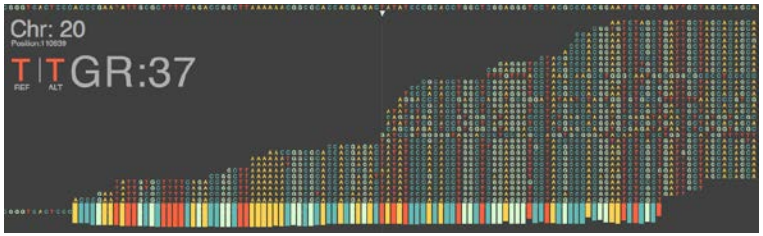


Gene Networks

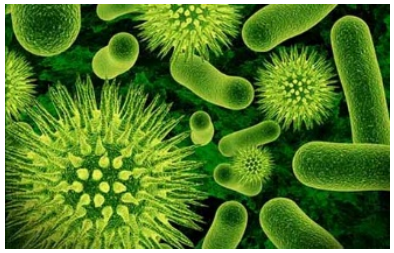
[HiPC 2015
SC 2014
IPDPS 2014
SC 2012
ICPP 2011
ICPP 2009
HiPC 2008]

Genome Assembly

[IPDPS 2010
ICPP 2008
IPDPS 2006]



Metagenomic Clustering & Assembly

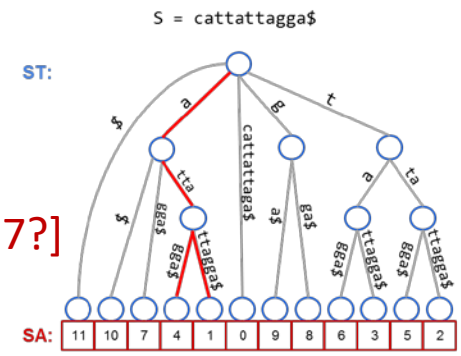


[SC 2015
IPDPS 2011]

Compression

[SC 2015
IPDPS 2017?]

Indexing

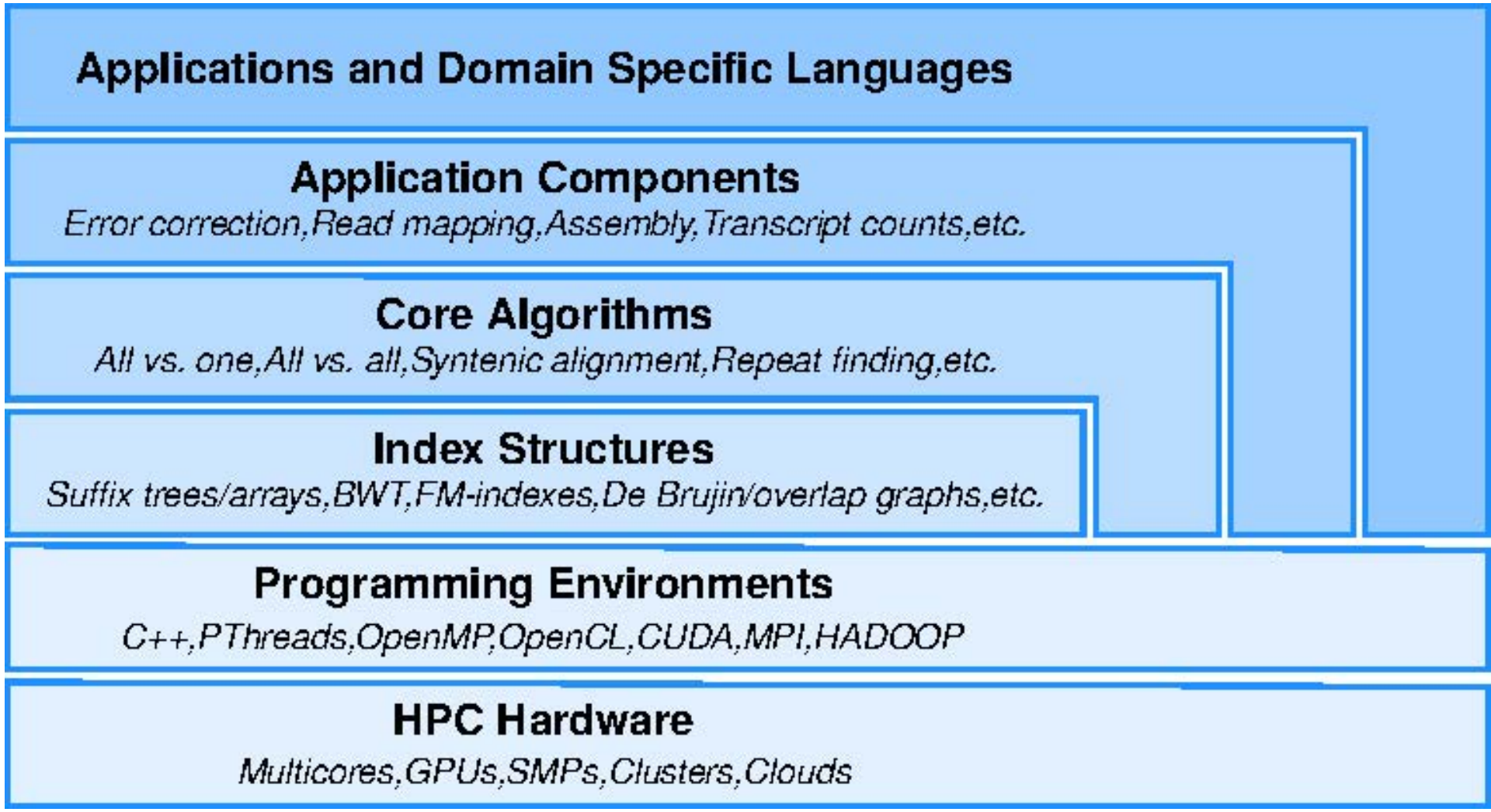


Pattern Matching

[SC 2016
IPDPS 2016
IPDPS 2014]

Genome Analysis

Genomes Galore – Big Data Analytics for High Throughput DNA Sequencing



k-mer Indexing

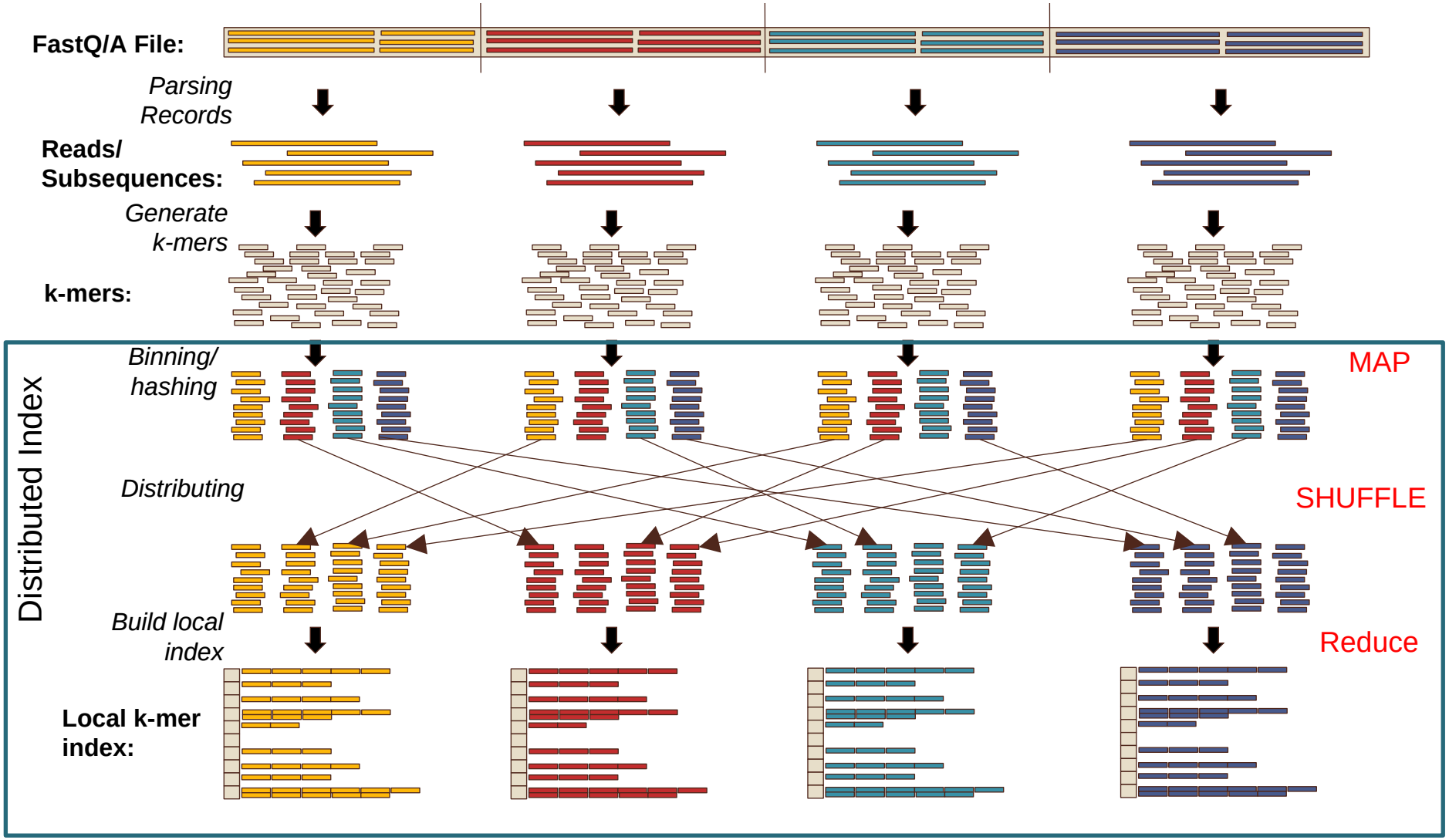
- *k*-mer: a length *k* substring of a DNA/RNA sequence
- *k*-mer indices are used to track the frequencies, positions, and quality scores of *k*-mers in the sequence data
- Commonly used by
  - Sequence alignment
  - Genome assembly
  - Error correction
  - Repeat detection
  - SNP detection
  - Comparative genomics
  - etc

Assembly	*k* <= 31	SoapDenovo2, Abyss
Error Detection / Correction	*k* ~= 17, 25	SoapDenovo2, AllPaths-LG
Repeat detection, alignment seeds	*k* ~= 100	AllPaths-LG

Kmerind Library

- Distributed memory *k*-mer index library
- Fully customizable and extensible
  - Key: *k*-mer or a transformation of a *k*-mer (e.g. reverse complement)
  - Value (user specified): read id, position in read, count/frequency, next base in read (i.e. de Bruijn graph edge), quality score, etc.
- Optimized for performance
  - For commonly used indices: *k*-mer counting and position indexing
- Provides standard index operations
  - *Insert*, *erase*, *find*, and *count* operations and their conditional counterparts
  - Building blocks for more complex operations on *k*-mer indices

Kmerind Algorithm



Kmerind: Performance

- Counting canonical DNA 31-mers in 7.5, 15, 30 GB metagenomics read sets, and human (3 GB) and pine (12 GB) whole genome, using 64 cores
- Both hashing and sorted array based Kmerind indices are faster than JellyFish 2, KMC 2, and Kmernator

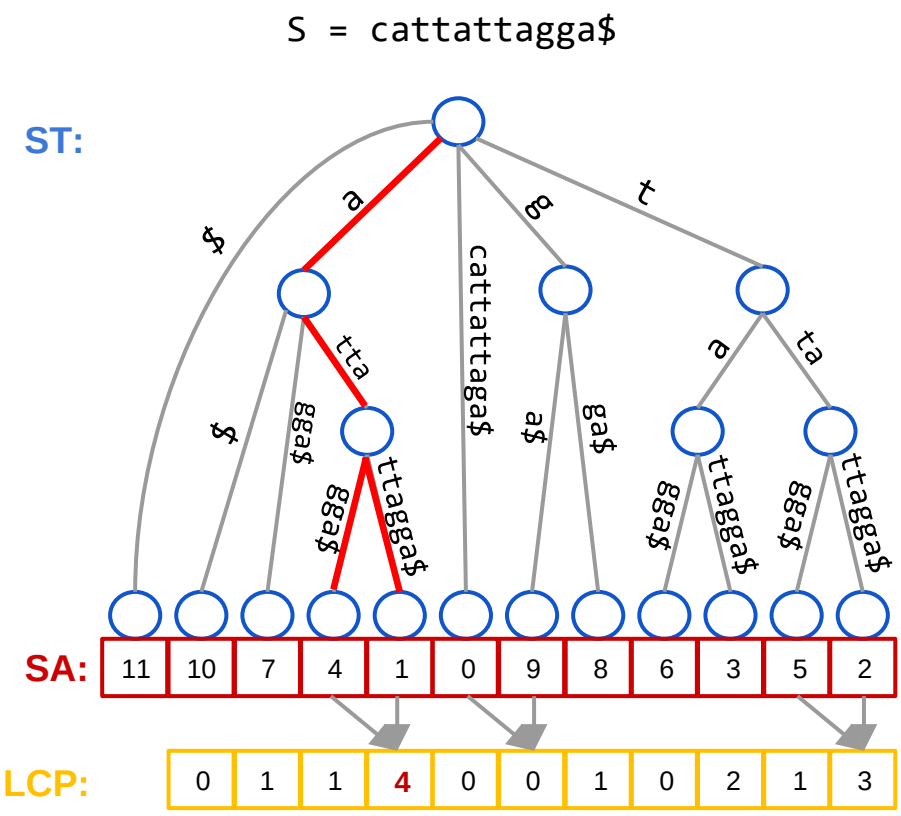
time (s)	Metagenomic Reads			Whole Genome	
	M1	M2	M3	G1	G2
JellyFish 2	36.27	67.35	84.46	20.60	66.13
KMC 2	23.44	48.19	83.28	115.63	341.35
Kmernator	84.00	172.00	349.00	–	–
Kmerind HASH	**9.97**	**20.04**	**42.52**	**13.04**	**50.98**
Kmerind SORT	12.20	24.96	50.15	15.19	61.19

Out-performs Kmernator on distributed memory by 6-8X

Pan et al., *ACM BCB* 2016

Parallel Suffix Arrays and LCP Arrays

- **Suffix Tree (ST)**
  - trie of all suffixes of a set of sequences
- **Suffix Array (SA)**
  - array of sorted suffixes
  - represented by their offset
- **Longest Common Prefix (LCP)**
  - length of prefix match between consecutive suffixes in **SA**
- **SA+LCP** equivalent to **ST**
  - for many applications
  - more space efficient but slower



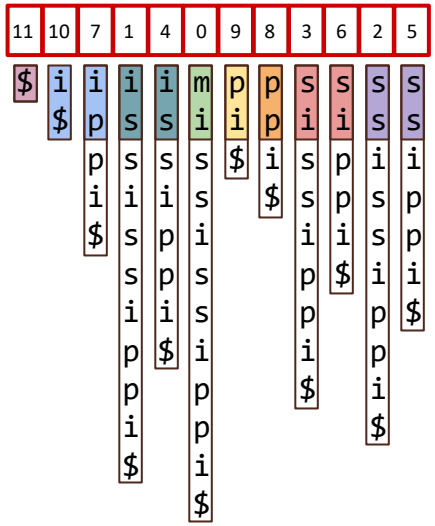
Prior State-of-the-art

- Genomic data sets are large
  - Human Genome 3.2 GB
  - Pine Genome 12 GB
  - Metagenomics 100 - 1000 GB (raw reads)
- Previous methods are limited, even for Human Genome:
  - External memory: Takes hours
  - Shared memory parallel:
    - Can't scale to large inputs (run out of memory)
    - *[Shun SC'14, Shun TOPC'14]:* ***SA****+****LCP****:* 105s, **ST**: 168s
  - Distributed memory parallel:
    - **Suffix Tree**: > 7 minutes *[Comin 2013]*
    - **Suffix Array**:
      - Not scalable to large inputs
      - Require input sequences to be in-memory on all processes

Parallel Suffix Arrays and LCP Arrays

- **Prefix Doubling**

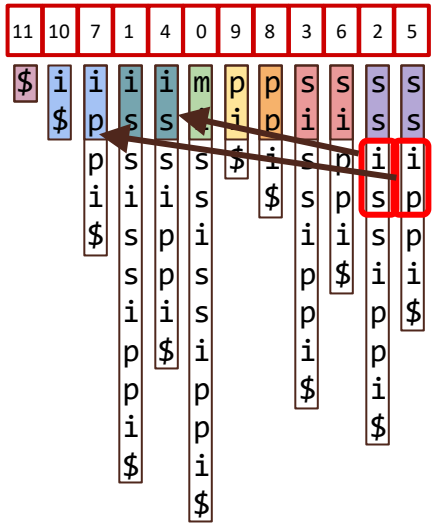
Sorted by prefix 2



Parallel Suffix Arrays and LCP Arrays

- **Prefix Doubling**

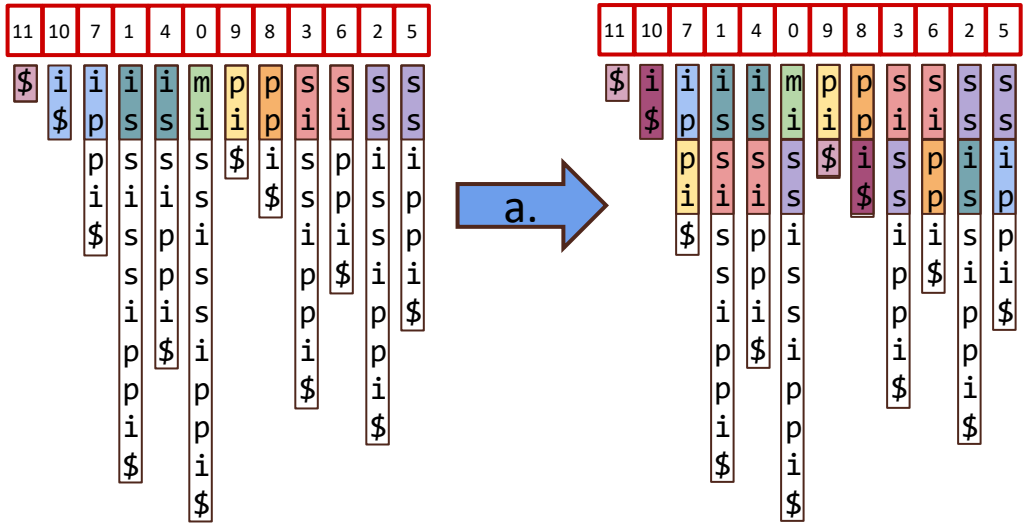
Sorted by prefix 2



Parallel Suffix Arrays and LCP Arrays

- **Prefix Doubling**

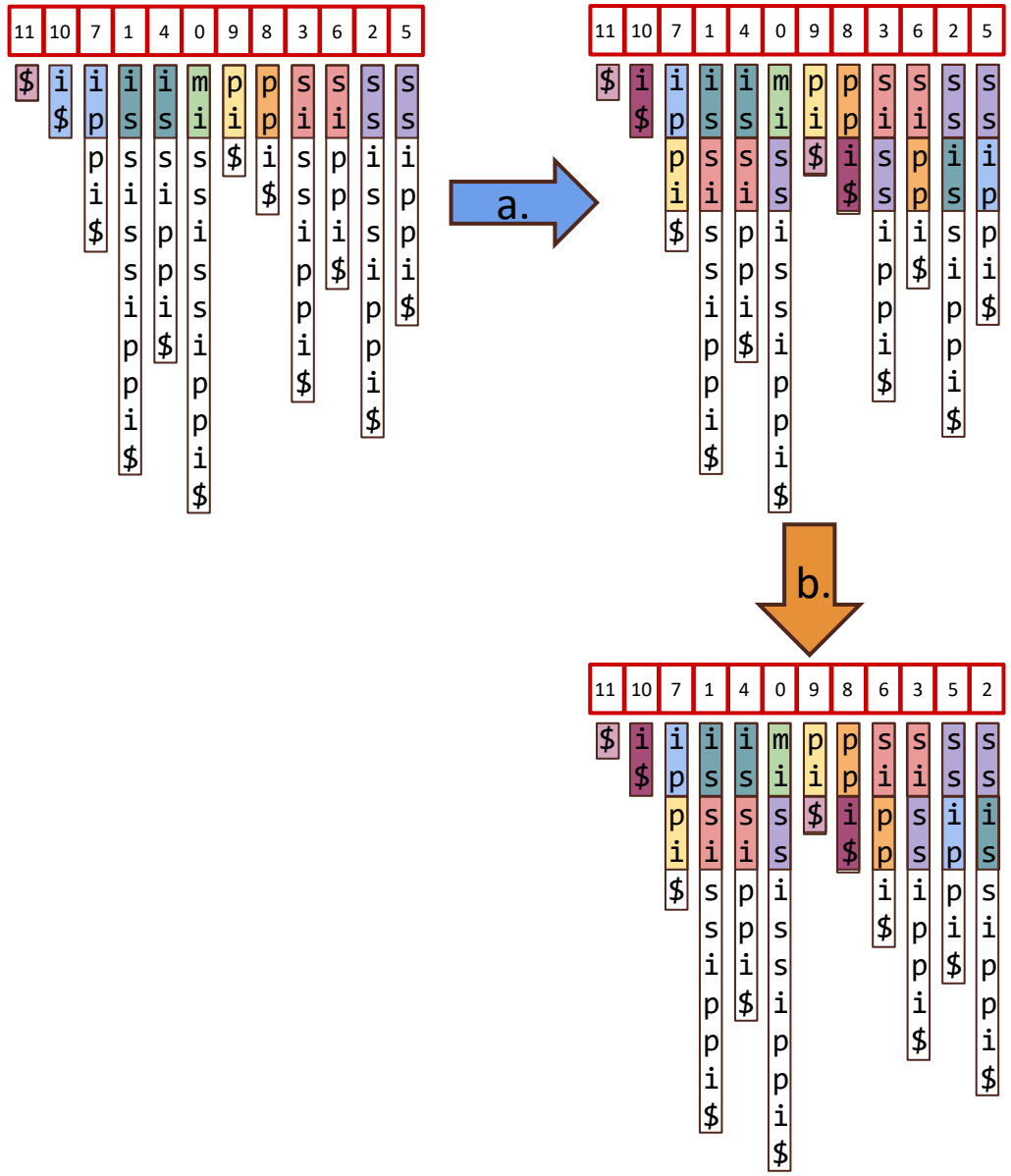
Sorted by prefix 2



Parallel Suffix Arrays and LCP Arrays

- **Prefix Doubling**

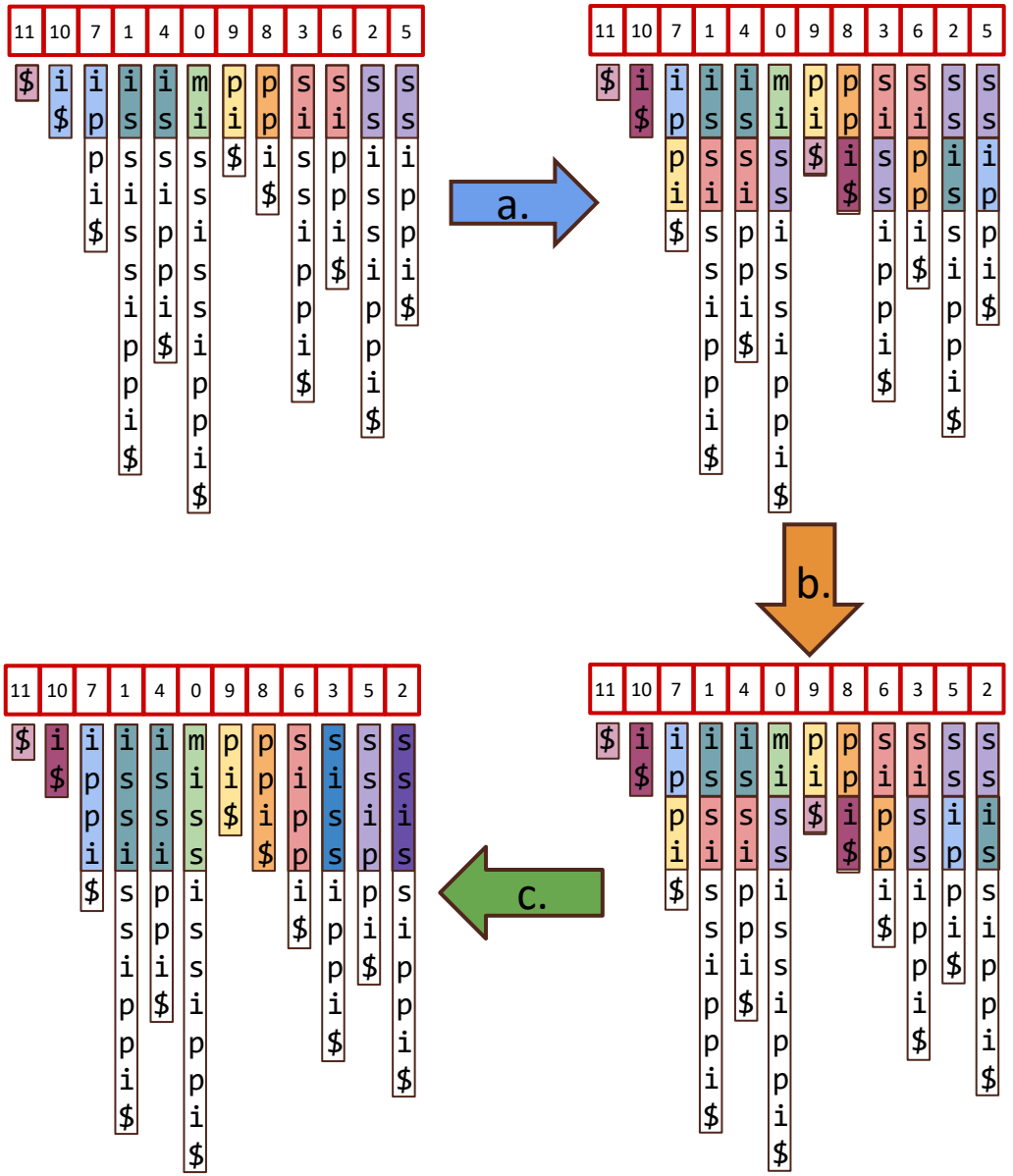
Sorted by prefix 2



Parallel Suffix Arrays and LCP Arrays

- **Prefix Doubling**

Sorted by prefix 2



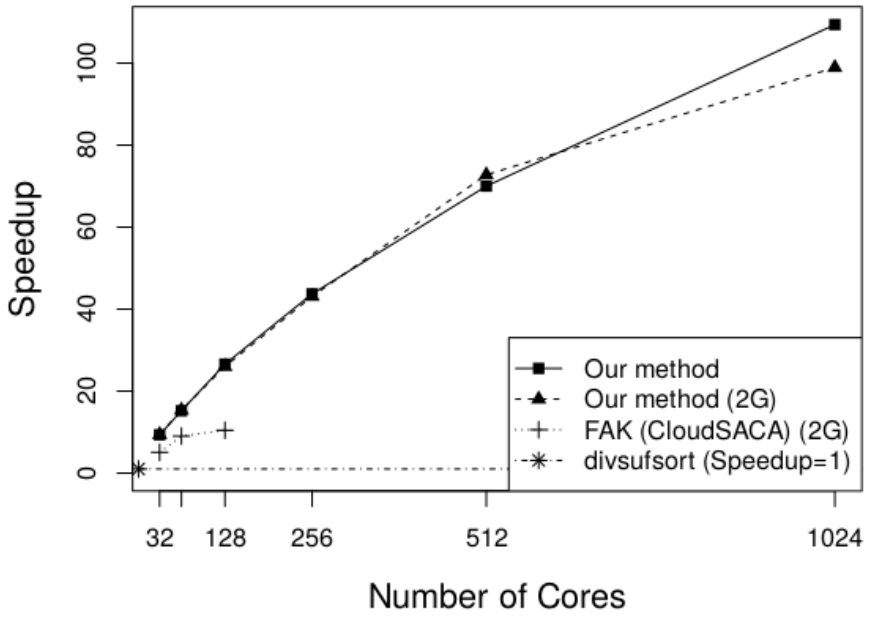
Sorted by prefix 4

Experimental Results for Suffix Arrays

Runtime (seconds) of different methods

Method	H 2G	H 3G	P 12G
divsufsort	424.5	586.4	X
mkESA (1)	586.6	1,123	X
mkESA (4)	462.6	759	X
cloudSACA (128)	40.6	X	X
Our method (128)	16.3	22.1	142.6
Our method (1600)	**3.5**	**4.8**	**14.8**

Speedup over divsufsort



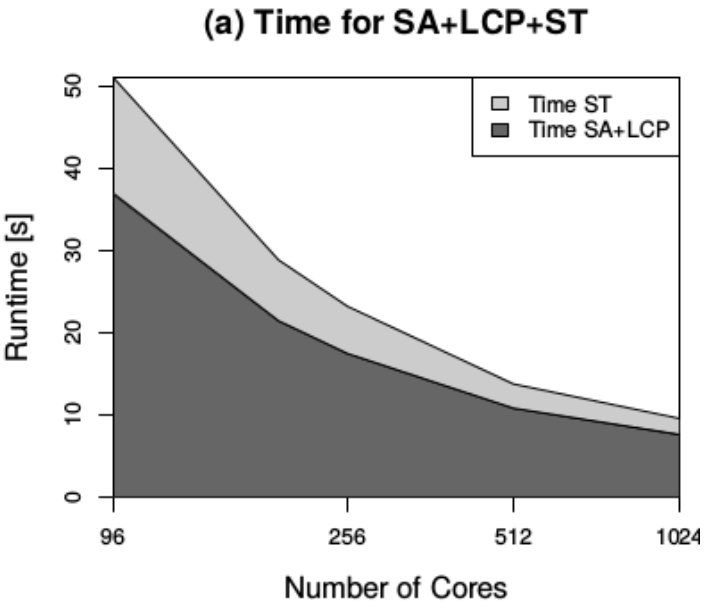
Experimental System:

- 100 nodes: 2x 8 core **Intel E5-2650**
- 128 GB RAM per node
- QDR Infiniband

Flick et al., *Supercomputing* 2015 Best Student Paper

Experimental Results for Suffix Trees

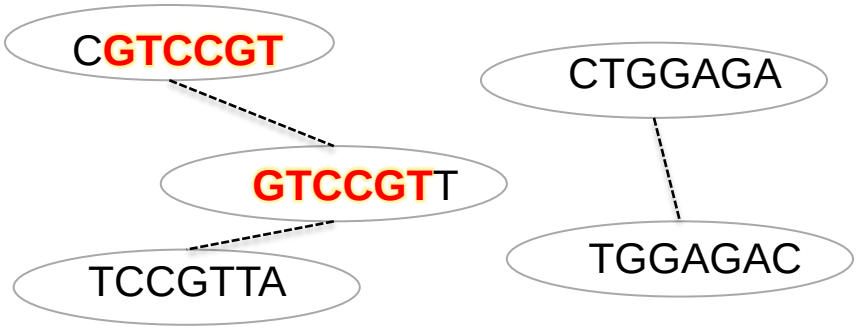
Method	System	Cores	Time
WaveFront	IBM BG/L	1024	15 min
ERA	16x Intel 2-core nodes	32	13.7 min
PCF	MareNostrum	172	7 min
Shun	4x 10 core **Intel E7-8870**	40	168 s
Shun	4x 18 core **Intel E7-8870**	72	146 s
Our method	4x 18 core **Intel E7-8870**	72	**63** s
Our method	64 nodes: 2x 8 core **Intel E5-2650**	1024	**9.5 s**



Flick & Aluru, IPDPS 2017 (under review)

De Bruijn Graph Partitioning

- Soil metagenomics dataset
  - Iowa Corn (1.8 billion reads)
  - Iowa Prairie (3.3 billion reads)
- High species-level heterogeneity
  - Disconnected components in de Bruijn graph (Howe *et al*. 2014)
  - 56 million components in Iowa Prairie dataset
  - 31 million components in Iowa Corn dataset

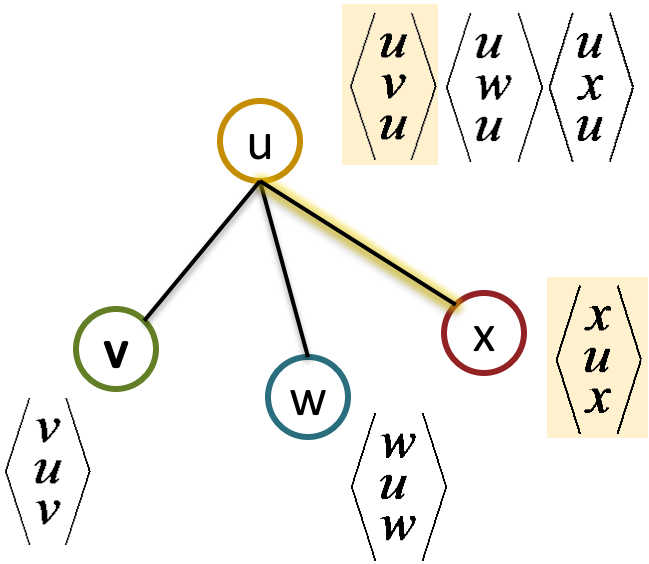


Parallel de Bruijn Graph Partitioning

- Distributed connected component labeling algorithm

Initialization

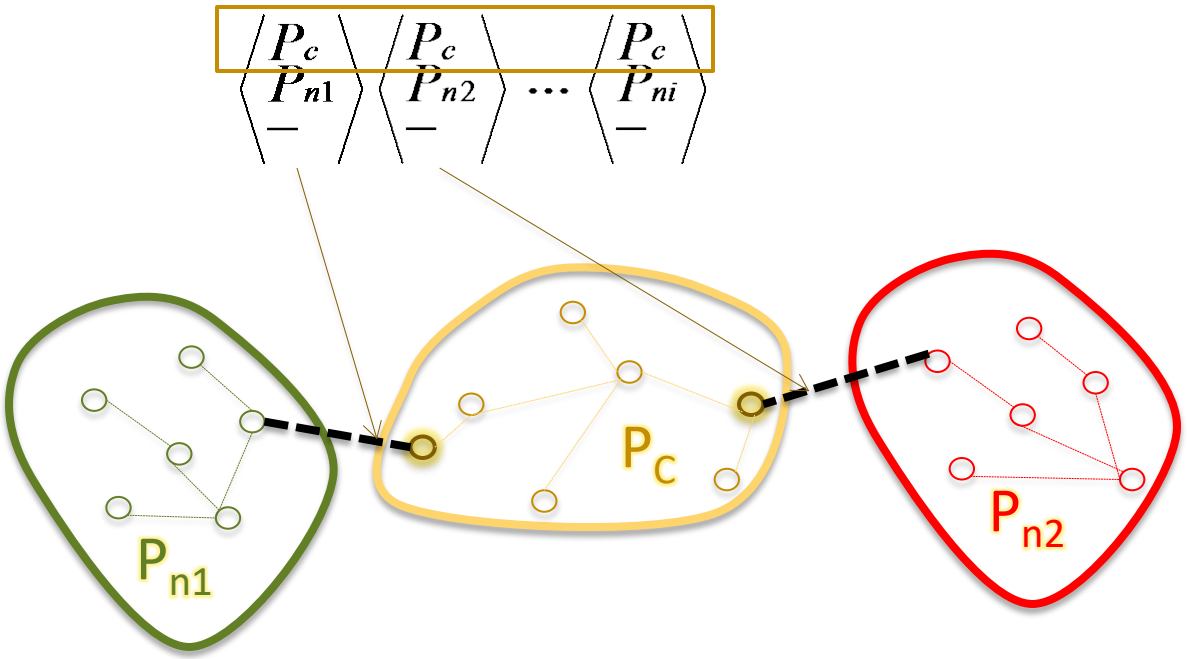
- Vector of tuples
- 2 tuples per edge in the graph
- Partition id of node = node id



⟨ Current partition id
Neighbor partition id
Node id ⟩

Parallel de Bruijn Graph Partitioning

- Each iteration
  - Do a parallel sort of all the tuples by *current partition Id*
  - Within each "bucket", compute minimum *neighbor partition id*
  - Flip the tuples to communicate my new partition id to neighbors in the next iteration

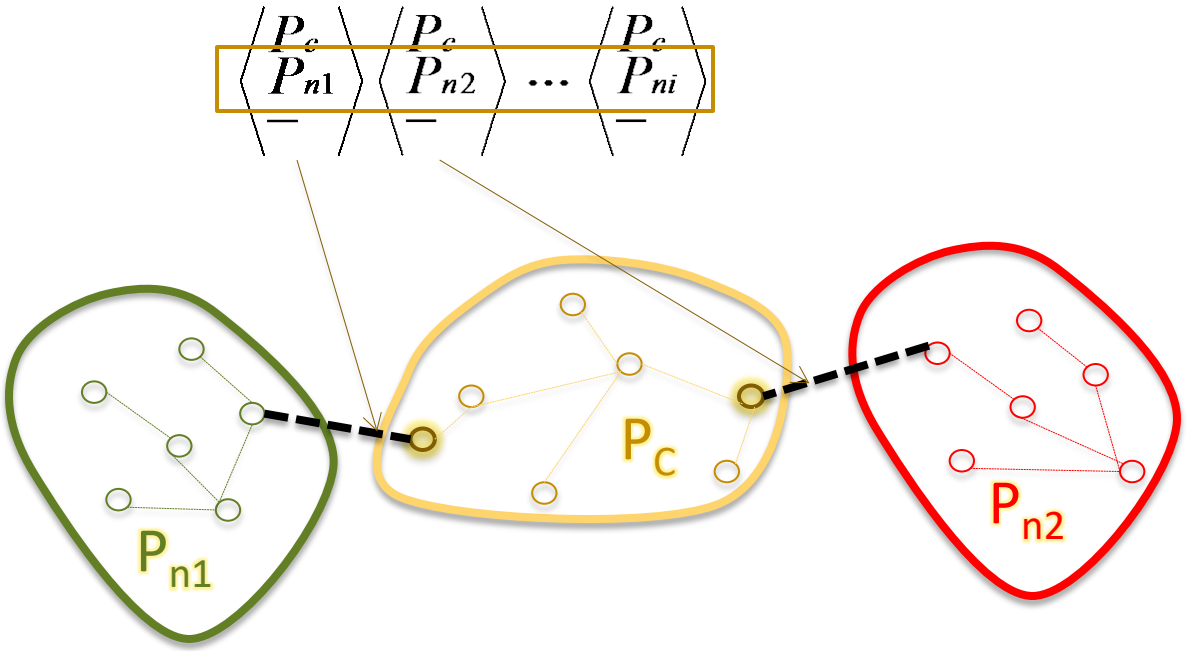


Parallel de Bruijn Graph Partitioning

- Each iteration
  - Do a parallel sort of all the tuples by *current partition Id*
  - Within each "bucket", compute minimum *neighbor partition id*

$$P_c' = \min(P_{n1}, P_{n2} \ldots P_{ni})$$

  - Flip the tuples to communicate my new partition id to neighbors in the next iteration

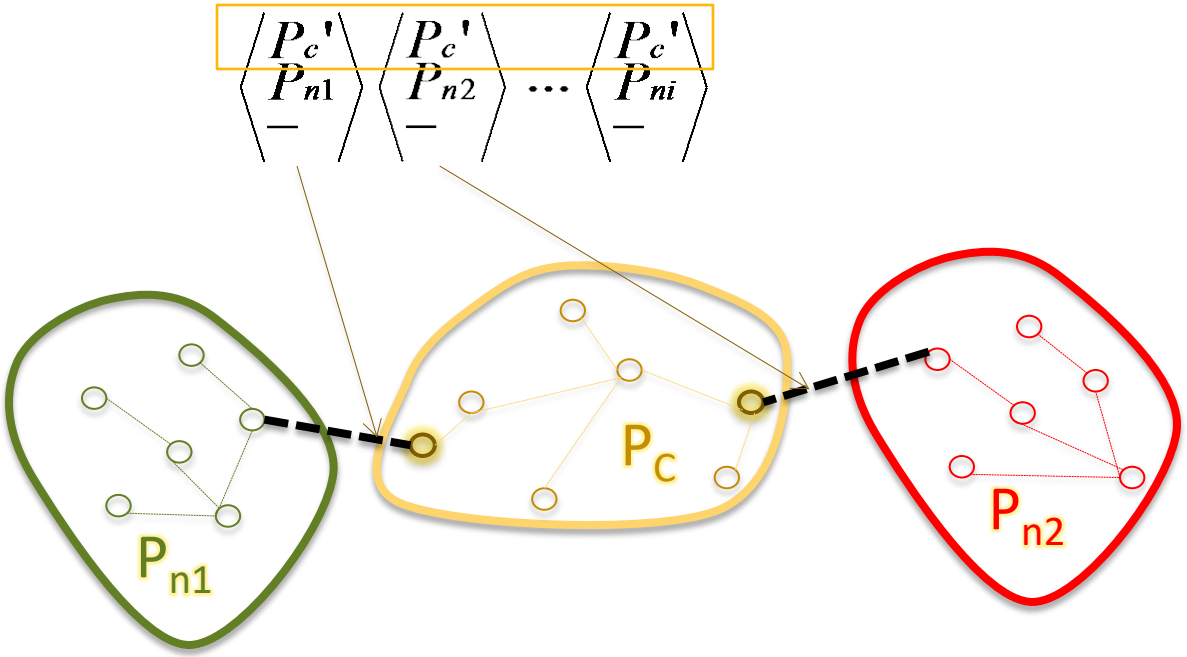


Parallel de Bruijn Graph Partitioning

- Each iteration
  - Do a parallel sort of all the tuples by *current partition Id*
  - Within each "bucket", compute minimum *neighbor partition id*

$$P_c' = \min(P_{n1}, P_{n2} \ldots P_{ni})$$
$$P_c \leftarrow P_c'$$

  - Flip the tuples to communicate my new partition id to neighbors in the next iteration

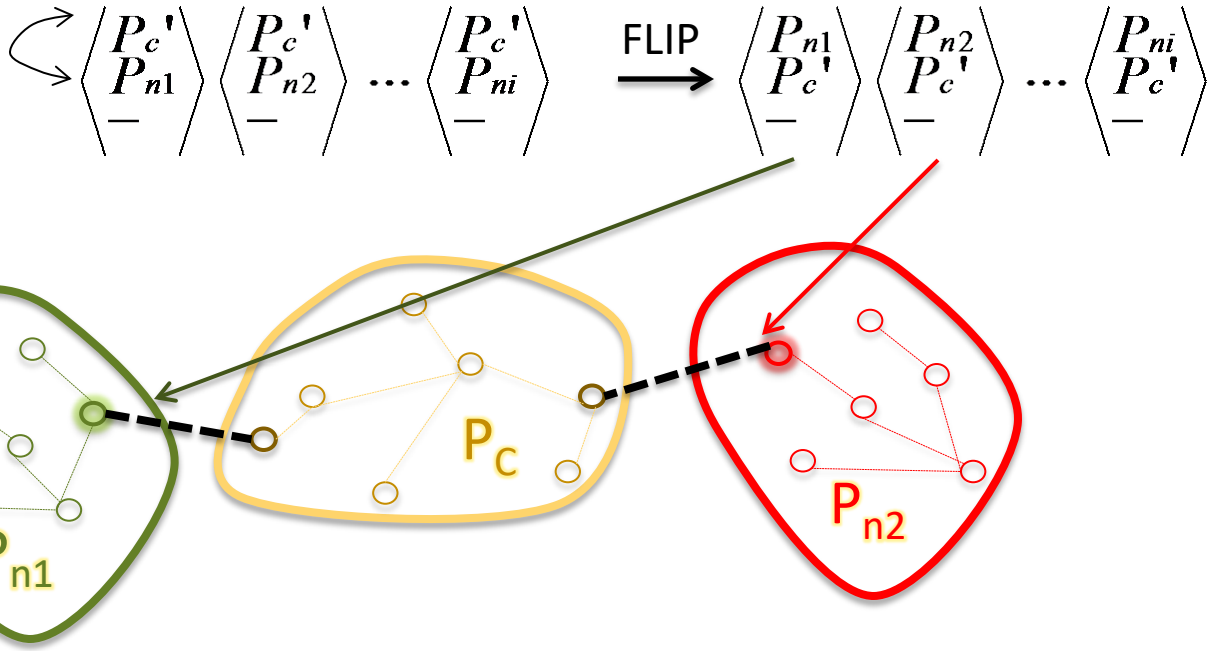


Parallel de Bruijn Graph Partitioning

- Each iteration
  - Do a parallel sort of all the tuples by *current partition Id*
  - Within each "bucket", compute minimum *neighbor partition id*

$$P_c' = \min(P_{n1}, P_{n2} \ldots P_{ni})$$
$$P_c \leftarrow P_c'$$

  - Flip the tuples to communicate my new partition id to neighbors in the next iteration



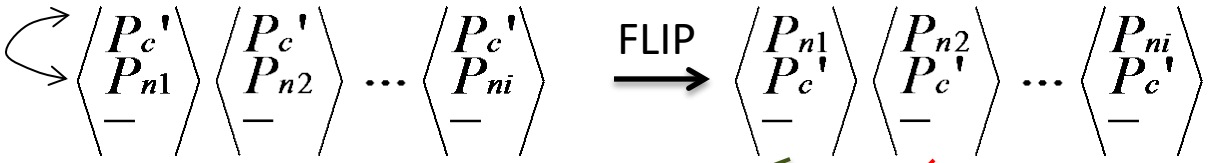
Parallel de Bruijn Graph Partitioning

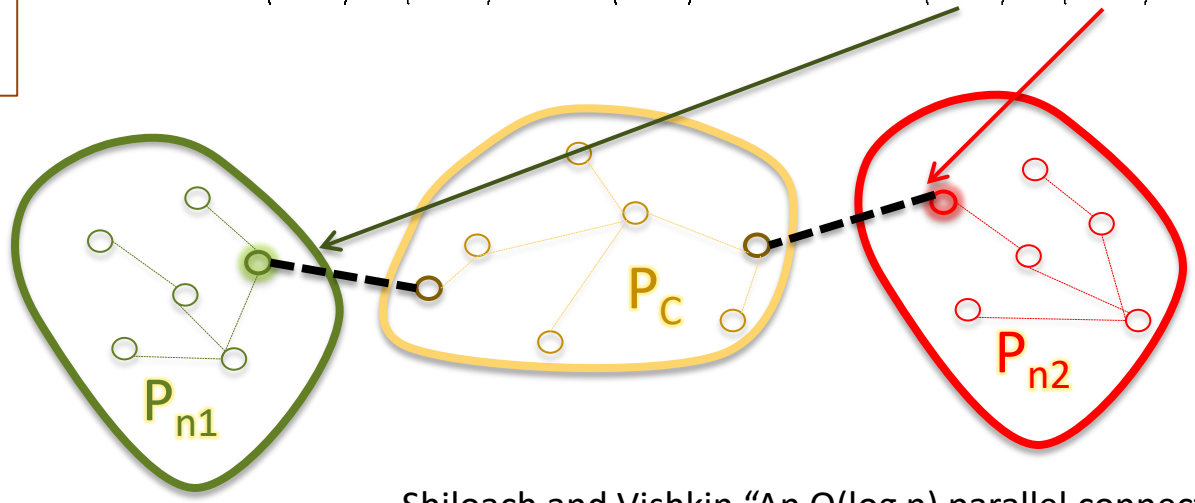
- Each iteration
  - Do a parallel sort of all the tuples by *current partition Id*
  - Within each "bucket", compute minimum *neighbor partition id*

$$P_c' = \min(P_{n1}, P_{n2} \ldots P_{ni})$$
$$P_c \leftarrow P_c'$$

  - Flip the tuples to communicate my new partition id to neighbors in the next iteration

Loop until convergence

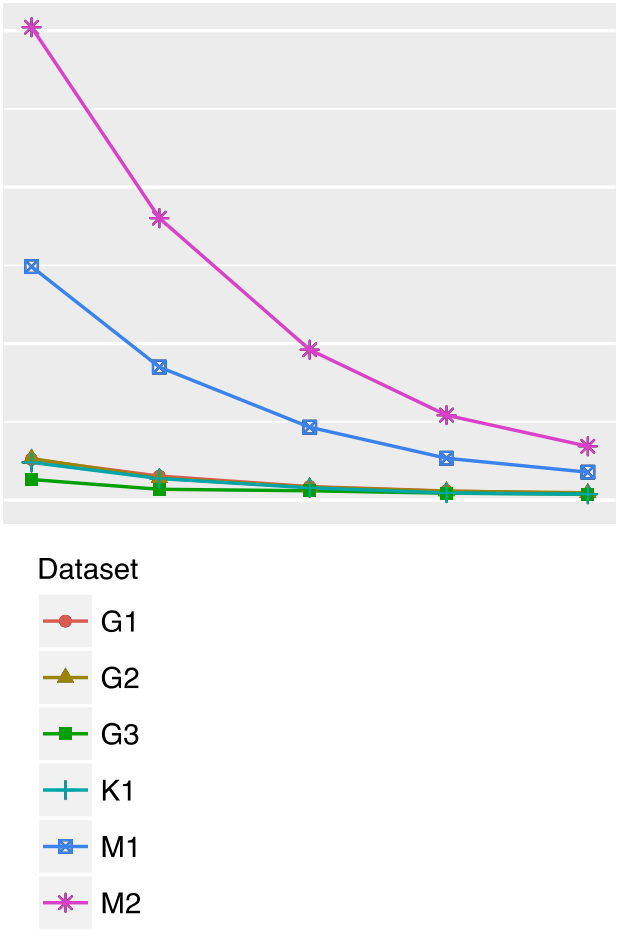




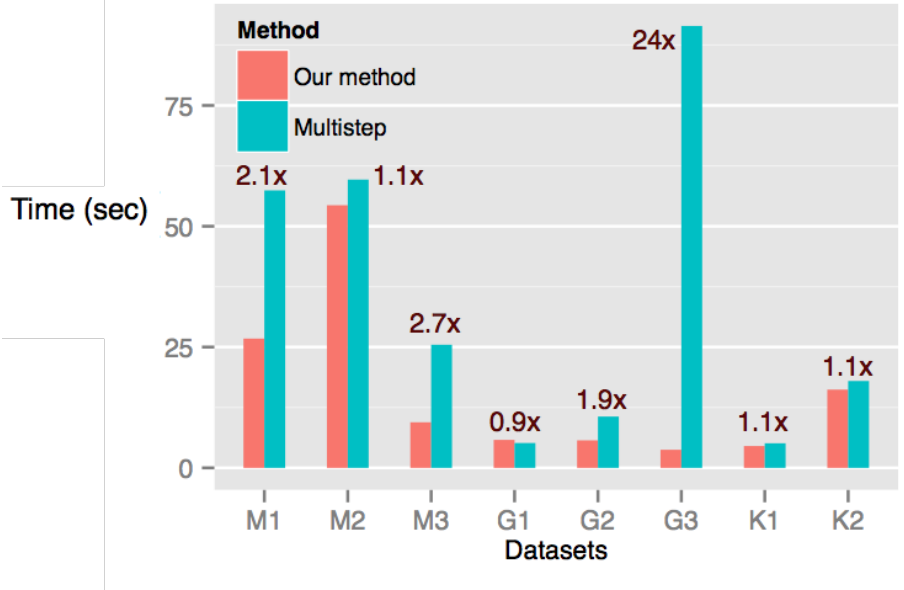
Shiloach and Vishkin "An O(log n) parallel connectivity algorithm" 1982

Parallel de Bruijn Graph Partitioning

Strong scalability up to 4,096 cores



- Partitioned graph with 54 billion edges in 3 minutes (for Iowa corn metagenomics dataset) using 32K cores
- Performance compares favorably against previous best distributed-memory connected components algorithm (Slota et al. IPDPS 2016).



Flick, Jain, Pan, and Aluru, *Supercomputing* 2015 (Reproducibility Initiative Winner – SC16)

Layer 2 – Core Algorithms

- Support common algorithmic constructs
- Two types of sequences
  - Fragments (reads: 100-300bp, 5000bp+)
  - Genomic Sequences (genomes, chromosomes, large genomic segments: 10^5 – 10^{10} bp)
- Computations within or across the two types of sequences

Layer 2 – Core Algorithms

- all vs. all between fragments
  - Common *k*mer
  - Spaced *k*mer (used in mapping)
  - Sharing a good fraction of *k*mers (Jaccard's coeff.)
  - Local alignment (exons within genes, etc.)
  - Suffix-prefix alignment (assembly)
- all (fragments) vs. one (genomic), all vs. few
  - return all mapping locations

Layer 2 – Core Algorithms

- Intra genomic
  - repeats, tandem repeats, CpG islands, retrotransposons, gene duplications
- genomic vs. genomic
  - SNPs, variation detection, large-scale genomic events
- Multiple genomic
  - Motif detection, gene content evolution, multi-gene synteny

Software Repository

Parallel Bioinformatics Library for Short Sequences (ParBLiSS) – github.com/ParBLiSS

Workshop on Parallel Software Libraries for Sequence Analysis (pSALSA)

- ACM BCB, September 2015, Atlanta
- IPDPS, May 2016, Chicago
- ACM BCB, October 2016, Seattle

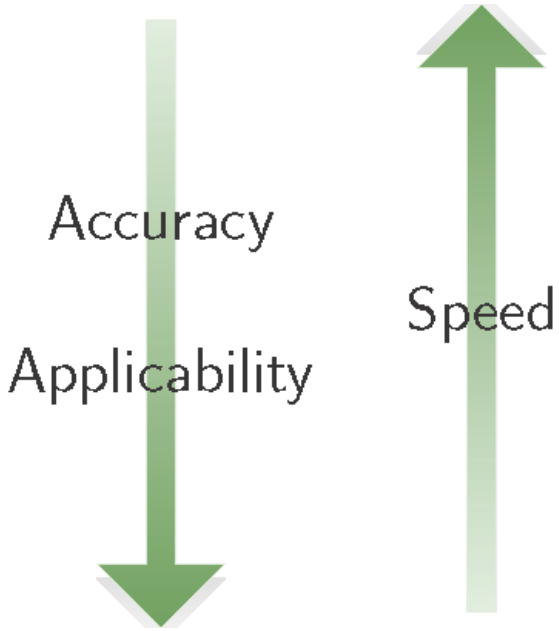
Whole-Genome Networks

- Arabidopsis Thaliana
  - Widely studied model organism
  - 125 Mbp genome sequenced in 2000
  - About 22,500 genes & 35,000 proteins
- NSF Arabidopsis 2010 Program launched in 2001
  - Goal: discover function(s) of every gene
  - ~$265 million funded over 10 years
  - Sister programs such as AFGN by German Research Foundation (DFG).
- Status today: a third of genes with no known function

- How can computer science help?
  - 11,760 gene expression datasets available in public databases.
  - Construct genome-wide networks to generate intelligent hypothesis.

Gene Networks

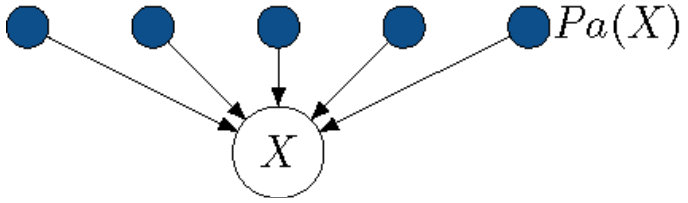
- Structure Learning Methods
  - Pearson correlation
    - (D'Haeseleer et al. 1998)
  - Gaussian Graphical Models
    - GeneNet (Schafer *et al*. 2005)
  - Information Theory
    - ARACNe (Basso *et al*. 2005)
    - CLR (Faith *et al*. 2009)
  - Bayesian networks
    - Banjo (Hartemink *et al*. 2002)
    - bnlearn (Scutari 2010)



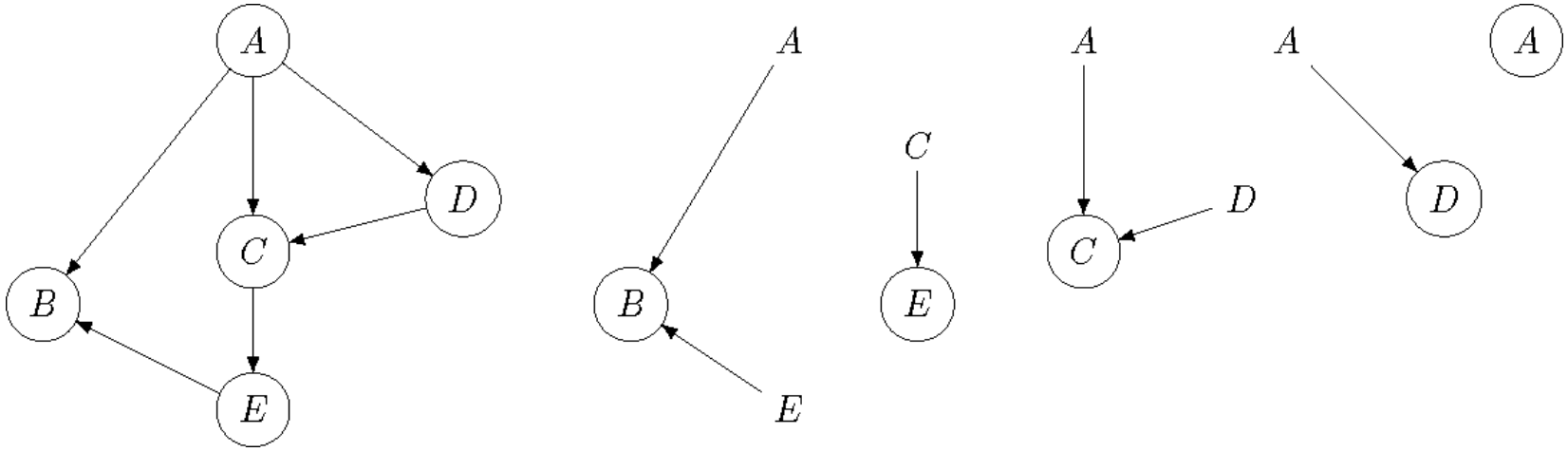
- Poor prognosis (Marbach *et al*. PNAS 2010)
  - Many do poorly on an absolute basis. One in three no better than random guessing.
  - Compromise: Quality of method vs. data scale

Score-based Bayesian Network Structure Learning

- Scoring Function: *s(X, Pa(X))*
  - Fitness of choosing set *Pa(X)* as parents of *X*



- Score of a network *N*



$$Score(N) = \sum_{X_i} s(Xi, Pa(X_i))$$

Parallel Heuristic Algorithm

1. Conservatively estimate candidate parent sets $CP(X)$ for each X
   - Symmetric: $Y \in CP(X) \implies X \in CP(Y)$
2. Compute optimal parents sets (OPs) from CPs using exact method
   - Directly compute *OP*s from small *CP*s ($CP(X) \leq t$)
   - Reduce large *CP*s by using

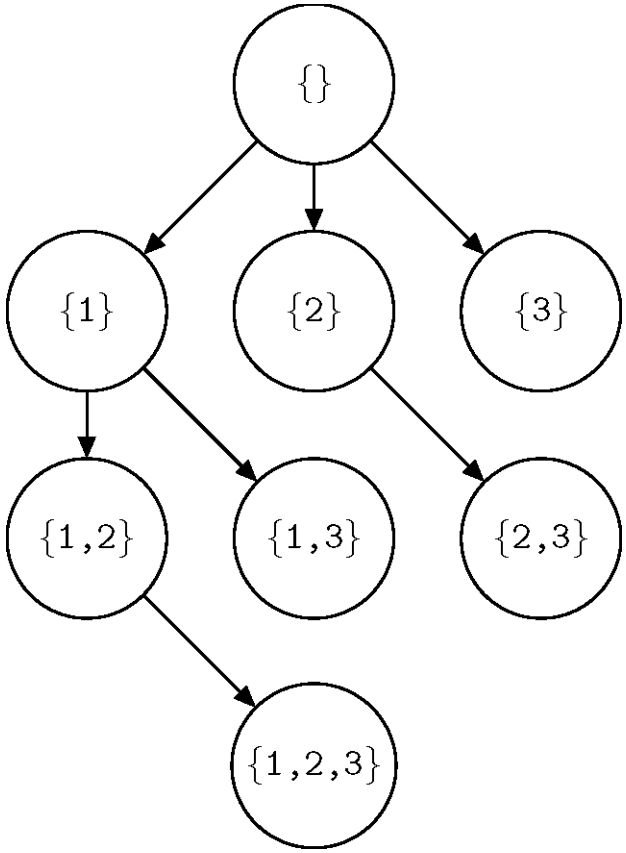
     $$CP(Y) \leftarrow CP(Y) \setminus \{X \in CP(Y) \mid Y \in OP(X)\}$$
   - Select top t correlations for still large *CP* sets
   - Directly compute *OP*s from the now small *CP*s
3. Detect and break cycles

Proposed Hypercube Representation

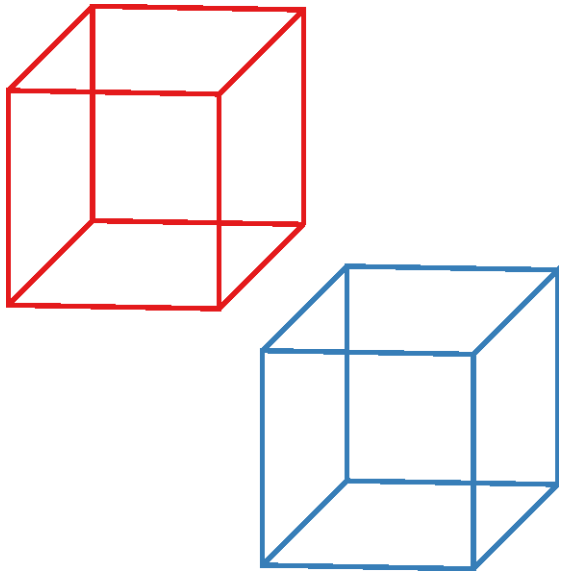
- Compute $CP(X) \rightarrow OP(X)$

$$OP(X_i) = \underset{A \subseteq CP(X_i)}{argmax}\ s(Xi, A)$$

- But, more efficient to compute $s(Xi, A)$ from $s(Xi, B)$ where $B \subset A$.
- Depth first traversal to cap memory usage



Work Decomposition



- Maximum unit of work is set as *r*-dimensional hypercube
- Larger hypercube are split into *r*-dimensional subhypercubes
- Direct access to subhypercube facilitated by directly computing the root.

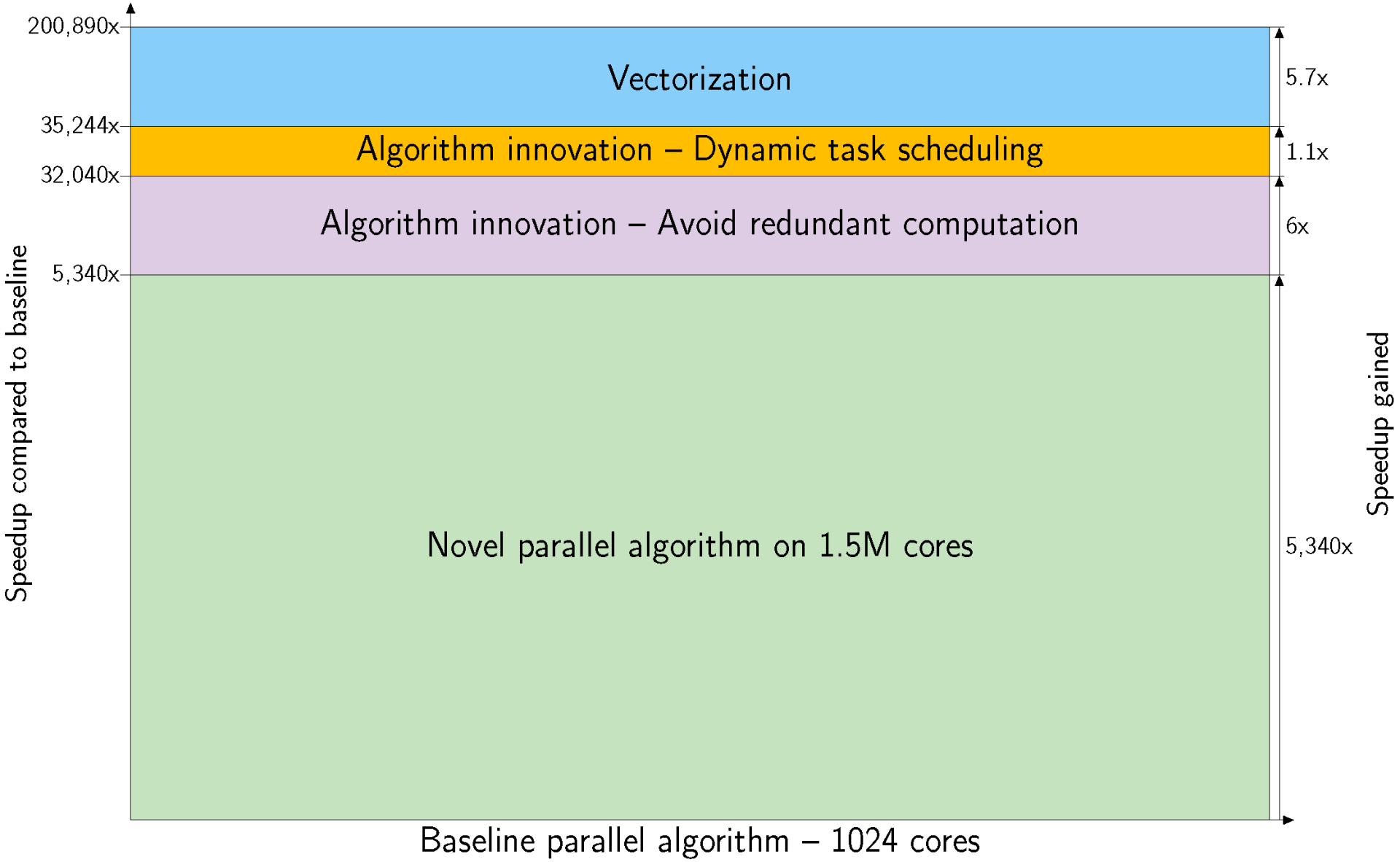
Key idea: Significantly increases parallelism with negligible compromise on reuse.

Target Supercomputers

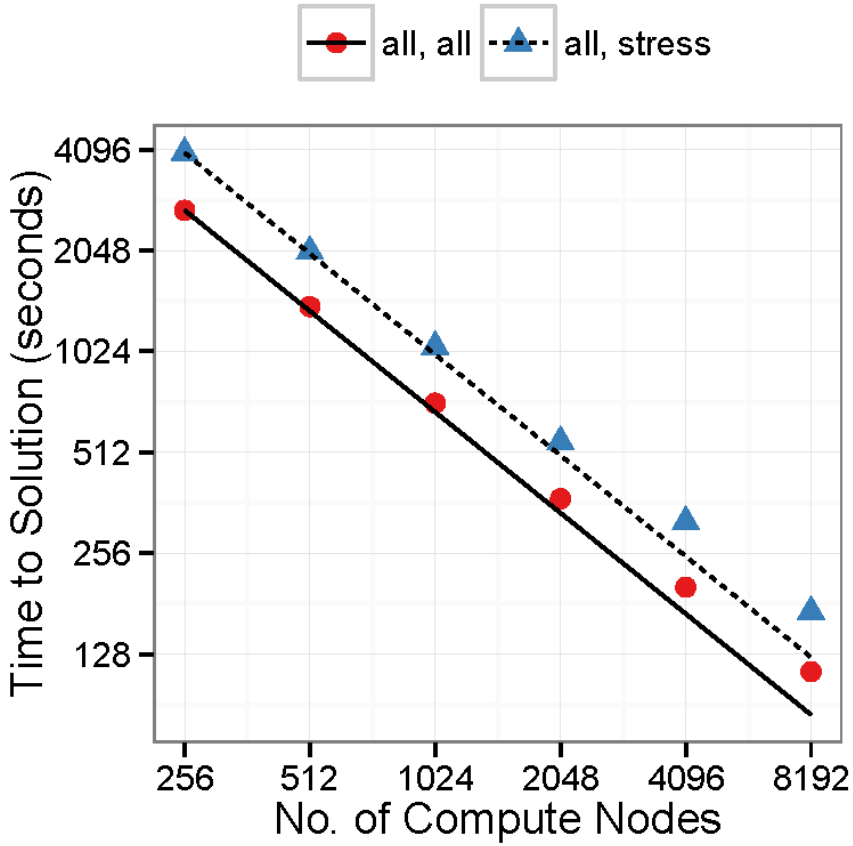
- Tianhe-2, National University of Defense Technology, Changsha.
- Stampede, Texas Advanced Computing Center, Austin

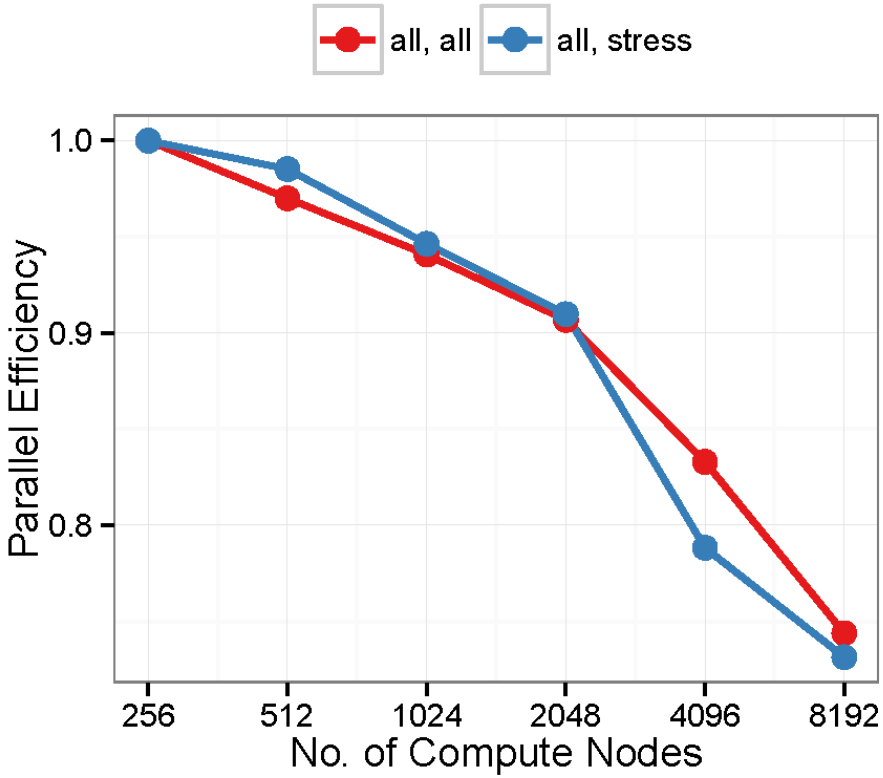
	Tianhe-2(54.9 PF)	Stampede(8.5 PF)
CPU	Intel Xeon E5-2600	Intel Xeon E5-2680
CPU Frequency	2.2 GHz	2.7 GHz
No. of CPUs	2	2
DRAM	64 GB	32 GB
Coprocessors	Intel Xeon Phi 31 S1P	Intel Xeon Phi SE10P
Coprocessors frequency	1.09 GHz	1.09 GHz
No. of Coprocessors	3	1
Coprocesssor Memory	8 GB	8 GB
Cores per node	192 (2 * 12 + 3*56)	76 * (2*8 + 60)
Threads per node	696	256

Where does speedup come from?



Parallel Efficiency

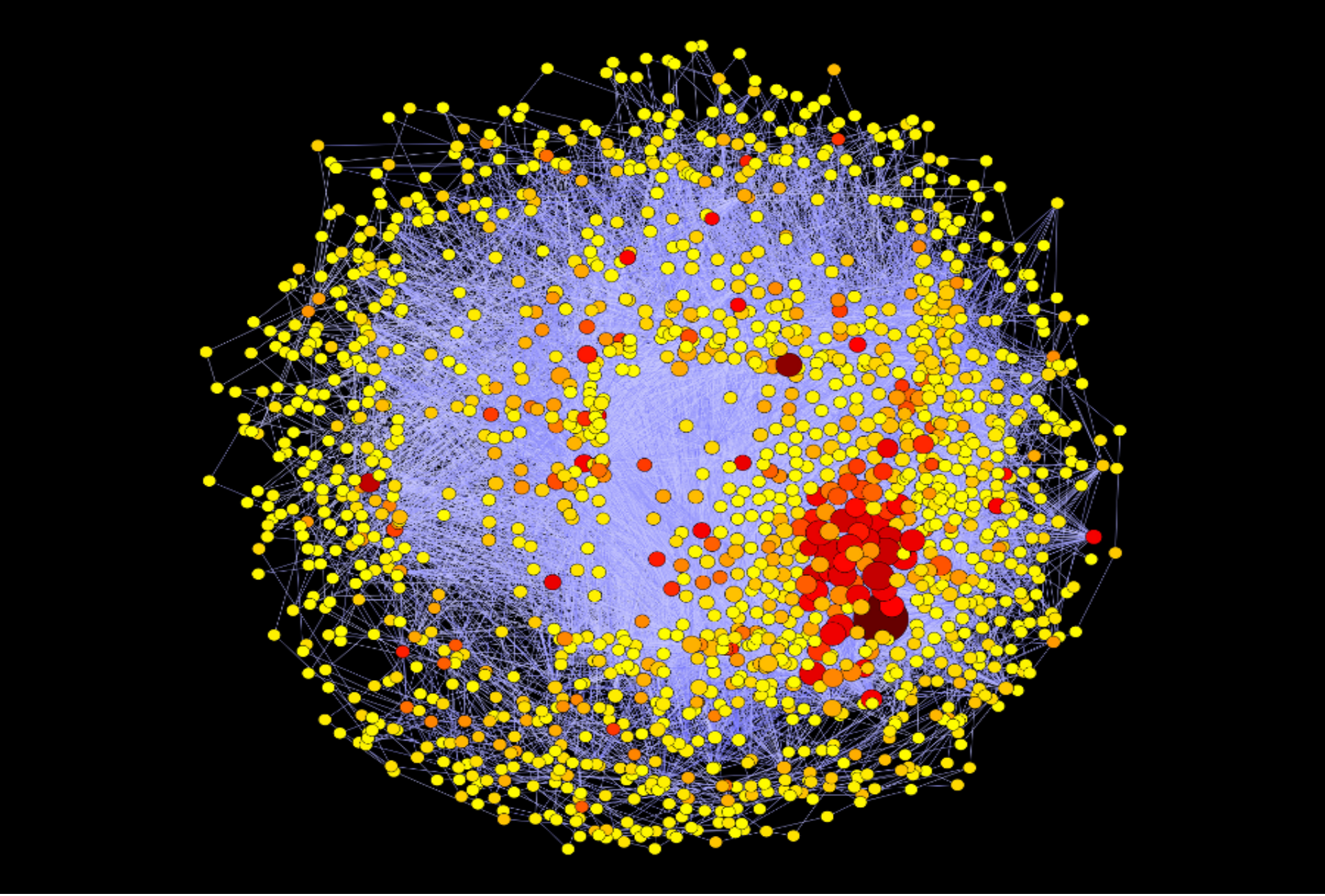




Misra et al., *Supercomputing* 2014
Best paper finalist

Full Application Runs

	all, all	seeding, all	root, all	all, stress
Genes(n)	14,330	13,590	15,236	15,216
Experiments(m)	11,760	4,933	1,939	2,476
Genes with |CP| <= t	13,922	13,086	14,340	13,293
Genes with reduced CP	408	504	896	1,923
Genes with truncated CP	241	15	293	1,376
Run-time on STP (sec)	1,941	269	501	2,352
Run-time on TH-2 (sec)	113.4			171.2
Billion scores/s (TH-2)	12.3			42.9



GeNA – Gene Network Analyzer

Adopted from page rank (Haveliwala, *IEEE Trans. Knowledge Data Engg.* 2003)

Assign transition probabilities:

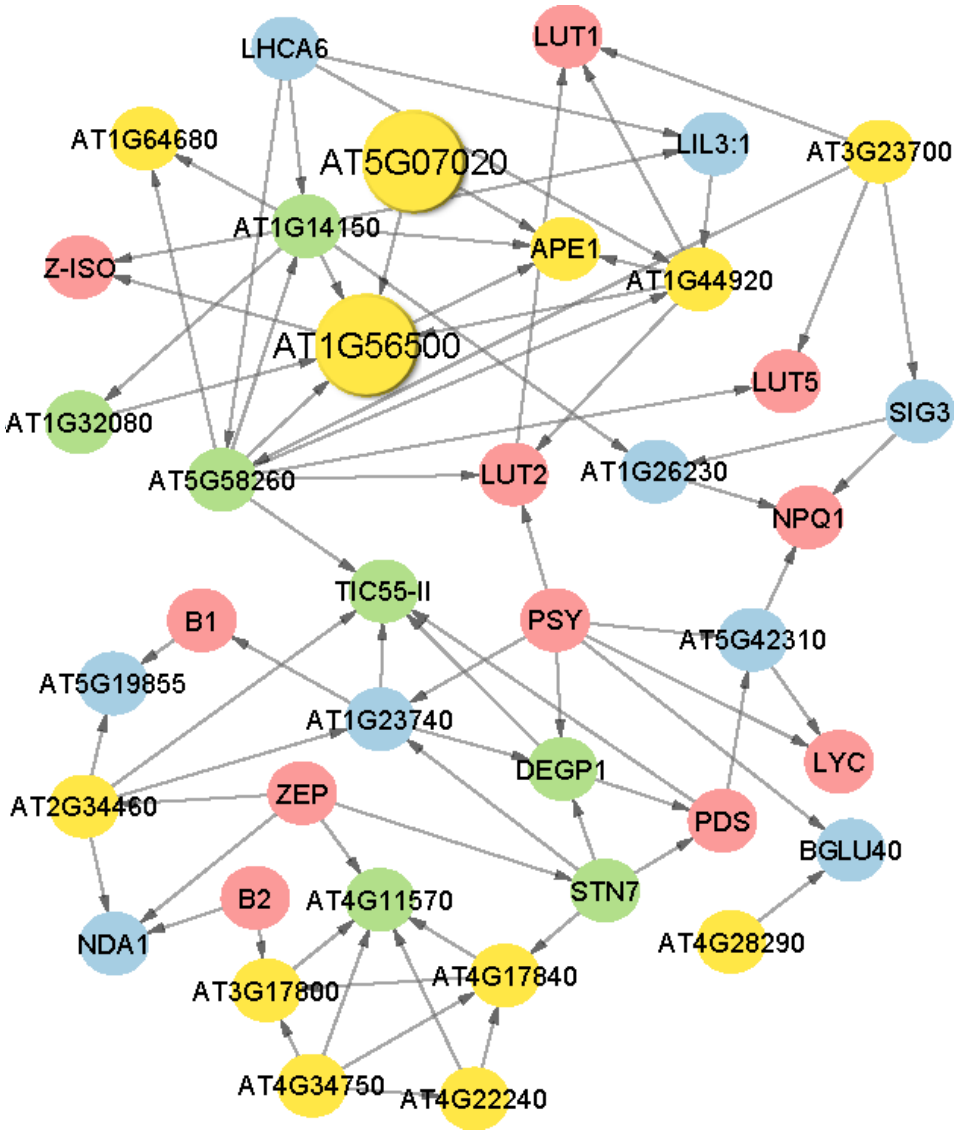
$$\omega(i,j) = \frac{(D[i,j])}{\sum_{k:(i,k)\in N} D[i,k]}$$

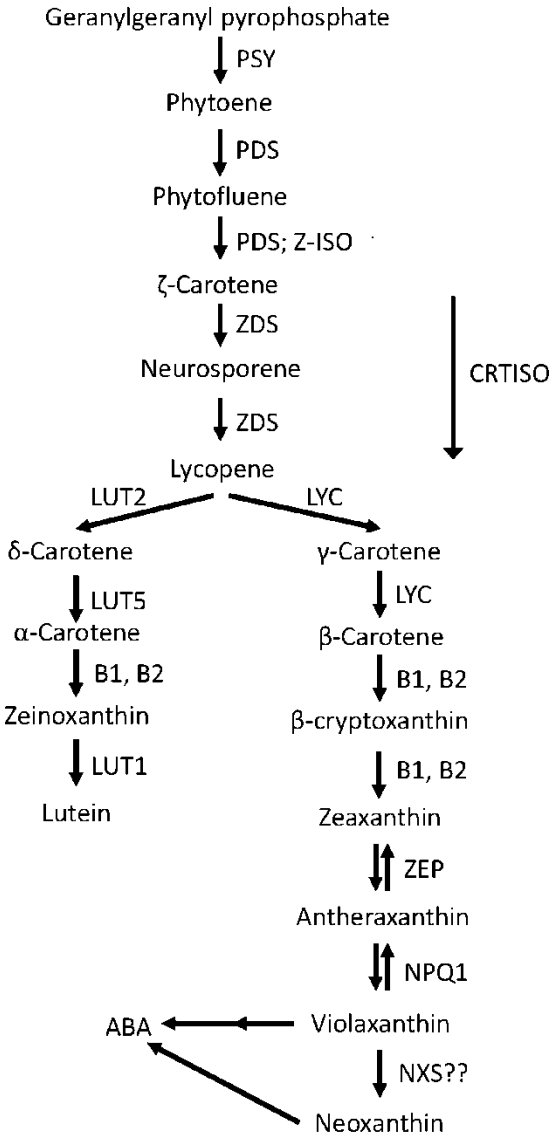
Compute ranks:

$$R(j)^{(k+1)} = (1-\alpha)\cdot\left(\sum_{i:(i,j)\in N} \omega(i,j)\cdot R(i)^{(k)}\right) + \alpha\cdot p(j)$$

Return connected subnetwork with high ranked genes

Carotenoid Subnetwork and Pathway

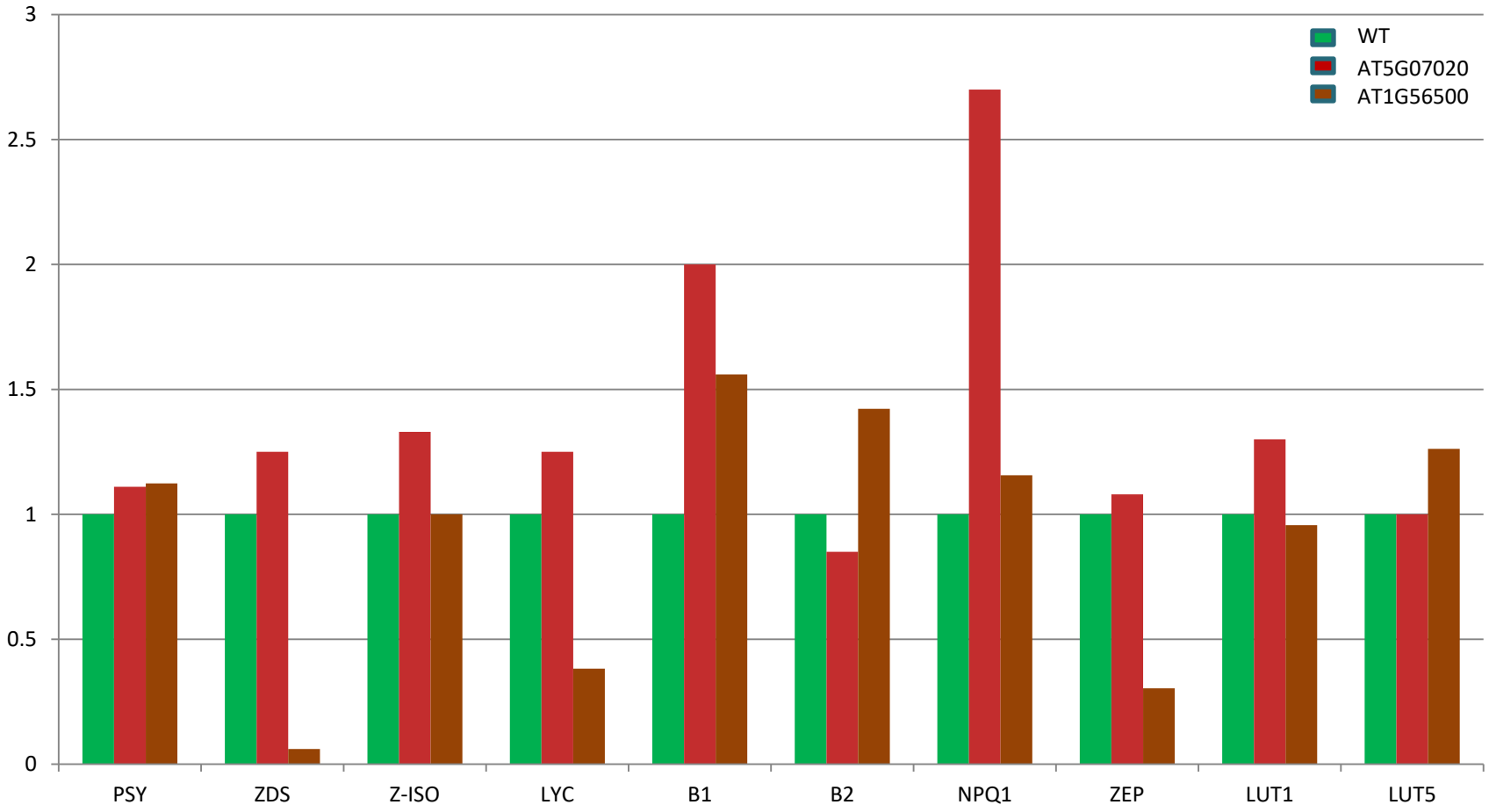




Arabidopsis T-DNA Mutants

Wild-Type **AT1G56500** **AT5G07020**

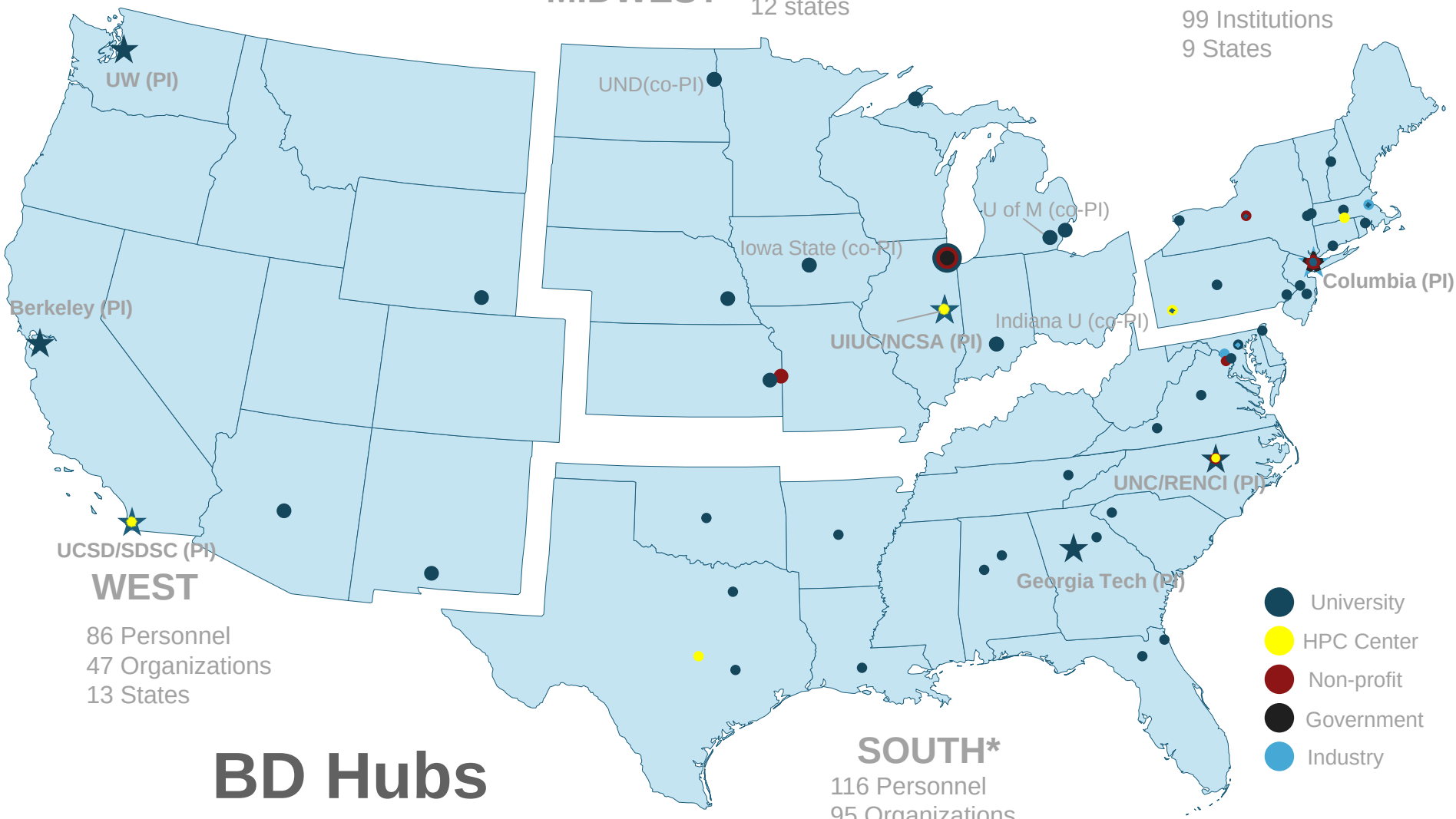
Carotenoid Genes – Expression Levels



NSF BD HUBS PROGRAM

1. Multi-stakeholder partnerships to go after regional/national/societal big data challenges
2. Set standards, share best practices
3. Provide data resources, develop infrastructure, create testbeds
4. Technology transfer, incubation
5. Education and workforce training
6. Engage with thought leaders, develop public awareness and ease adoption

Alaska & Hawaii are part of the West region
US Territories can participate in any region



BD Hubs

Points indicate affiliations of individuals named as steering council members and/or task leads.

*South points indicate Senior Personnel

Georgia Tech Institute for Data Engineering and Science

CREATING THE NEXT®

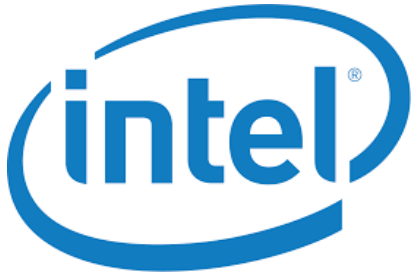
Acknowledgements

Current and Former Students:

- Sriram Chockalingam
- Scott Emrich
- Patrick Flick
- Benjamin Jackson
- Chirag Jain
- Nagakishore Jammula
- Ananth Kalyanaraman
- Rahul Nihalani
- Olga Nikolova
- Tony Pan
- Indranil Roy
- Abhinav Sarje
- Xiao Yang
- Jaroslaw Zola

Collaborators:

- Maneesha Aluru, GT
- Karin Dorman, ISU
- Wu-Chun Feng, V. Tech
- Kune Olukotun, Stanford
- Aditaya Ramamoorthy, ISU
- Patrick Schnable, ISU
- Charles Sing, U. Michigan



QUESTIONS?

Georgia Tech Institute for Data Engineering and Science