

Parallel Short Sequence Mapping for High Throughput Genome Sequencing

Doruk Bozdag, Umit Catalyurek

Dept. of Biomedical Informatics

Dept. of Electrical & Computer Engineering

The Ohio State University

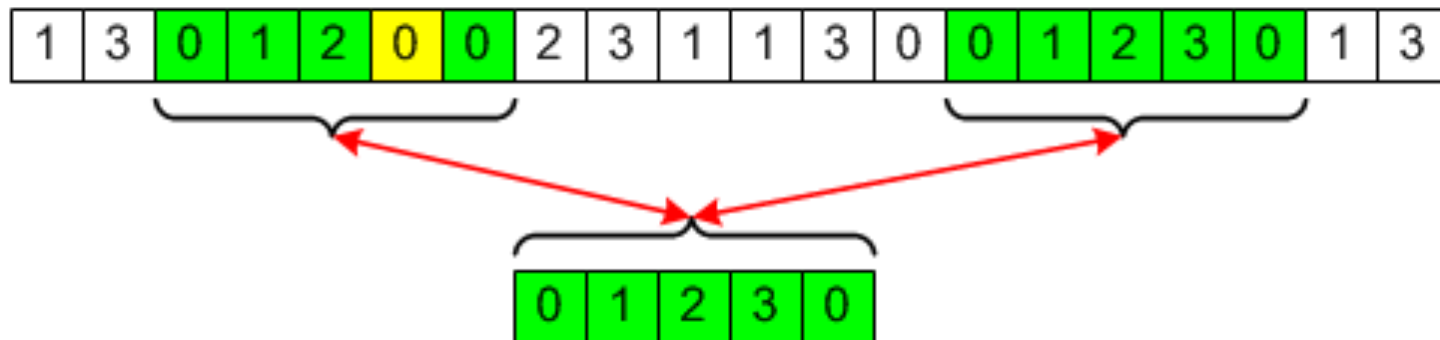
Catalin Barbacioru

Applied Biosystems

- Short Sequence Mapping Problem
- Related Work
- Covering Designs
- New Parallelization Methods
- Experimental Results
- Conclusion and Future Work

- Next generation sequencing instruments (SOLiD, Solexa, 454) can sequence up to 1 billion bases a day
 - 35-50 base reads
- Reads should be efficiently mapped to a reference genome
 - Human genome: 3 billion bases
- Sequentially mapping a single run (~130M reads, 3G base genome) takes about a day
- Fast, resource efficient, parallel algorithms that can handle mismatches are required

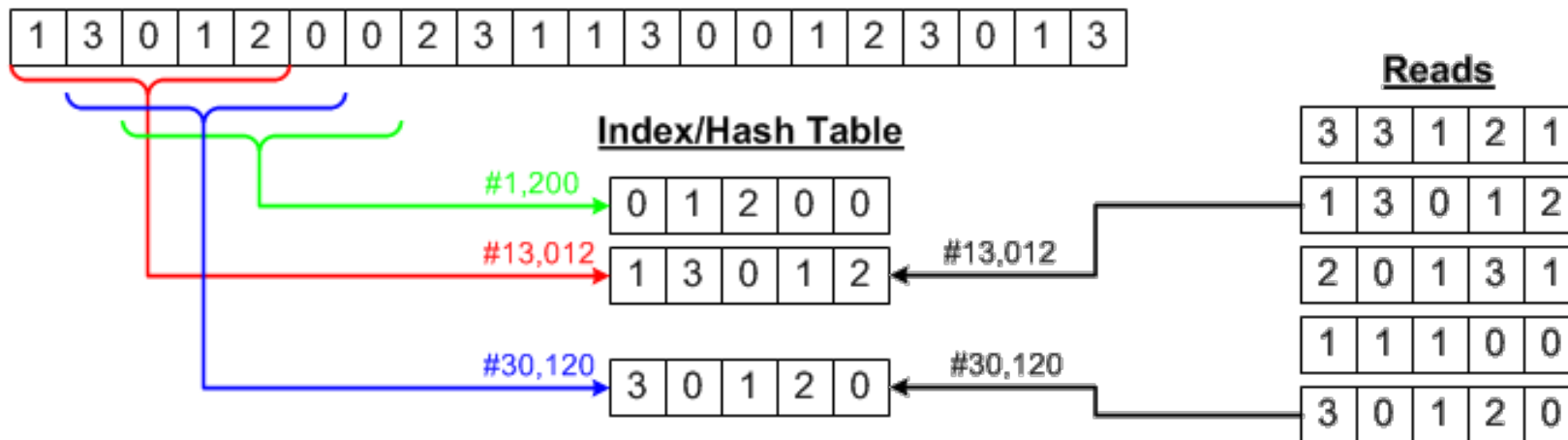
- Identify matching locations of short sequences (reads) on reference genome.
 - Allow mismatches



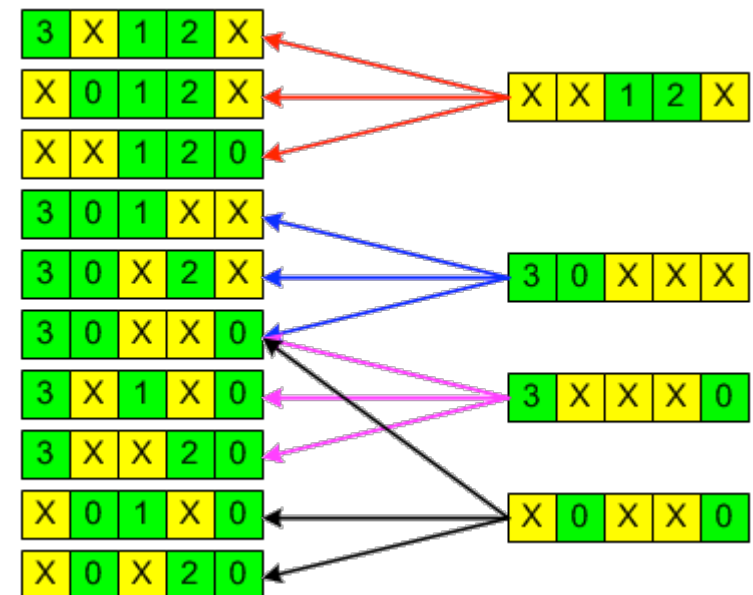
- Local sequence alignment
 - BLAST, PatternHunter
 - Start with a short sequence (anchor) mapping operation (14-18 bases long), then extend matches
 - No mismatch tolerance at anchor positions
 - PatternHunter improves sensitivity using spaced seeds
 - Target longer alignments and more general alignment problems
- Short sequence mapping
 - xMAN, Eland, PASS, SOAP, BLAT, SSAHA, MAQ, MosaicAligner, ZOOM, SHRIMP, MapReads.
 - Often a hash or an index table is utilized
 - Mismatches are either not allowed or handled by enumeration

Sequence Mapping Example

- Two step approach
 - Index/hash table construction using sliding window
 - Table lookup to find matches for each read



- Consider a sequence of length 5
- Checking for every possible 2-mismatch scenario requires $\binom{5}{2} = 10$ comparisons
- Find a set of 3-mismatch patterns to cover all possible 2-mismatch cases
 - Minimize the number of 3-mismatch patterns



Pros:

- Accounts for mismatches with full sensitivity
- Reduces the size of the hash table and the number of table lookups while matching a read
- Best known covering designs are publicly available (La Jolla Covering Repository)

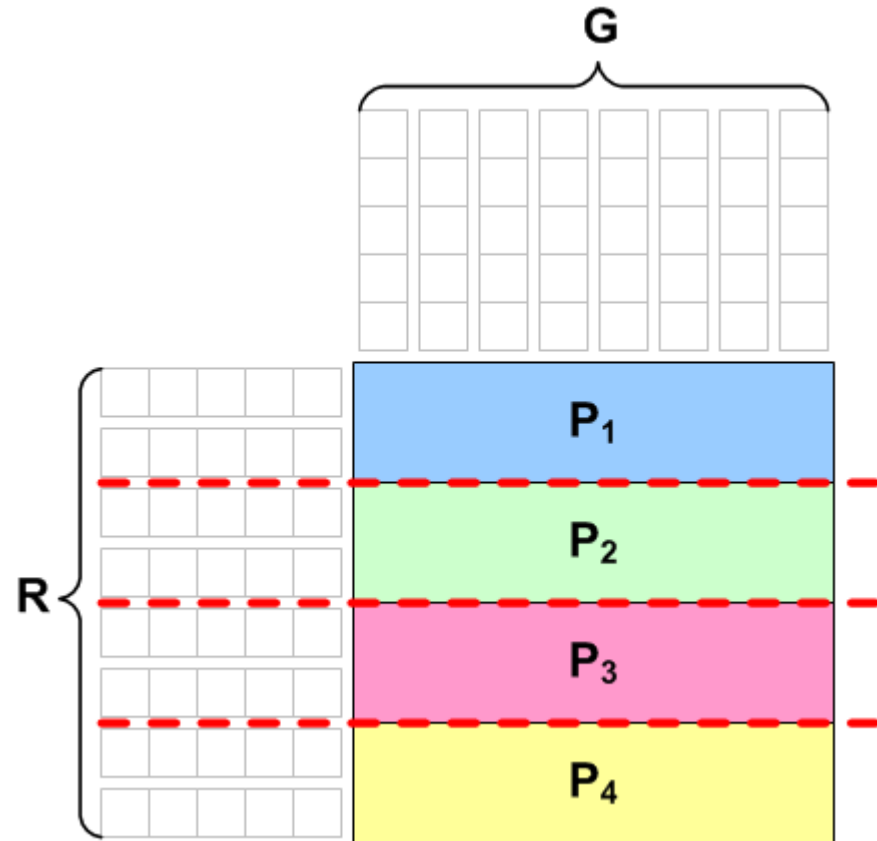
Cons:

- Requires post processing to remove hits having mismatches greater than the targeted number

- We propose 6 parallelization methods for hash/index based short sequence mapping
 - Performance depends on data size and number of nodes
 - **G**: Size of reference genome
 - **R**: Number of reads
 - **N**: Number of computation nodes
- Modeling unit computation cost
 - c_g : Time to compute an index/hash for a single sequence
 - In sequential algorithm, index table is constructed in $c_g G$
 - c_r : Time to process a single read if no collision
 - c_c : Average time to resolve a collision
 - In sequential algorithm, all reads are processed in $(c_r + c_c G)R$

Partitioning Reads Only (PRO)

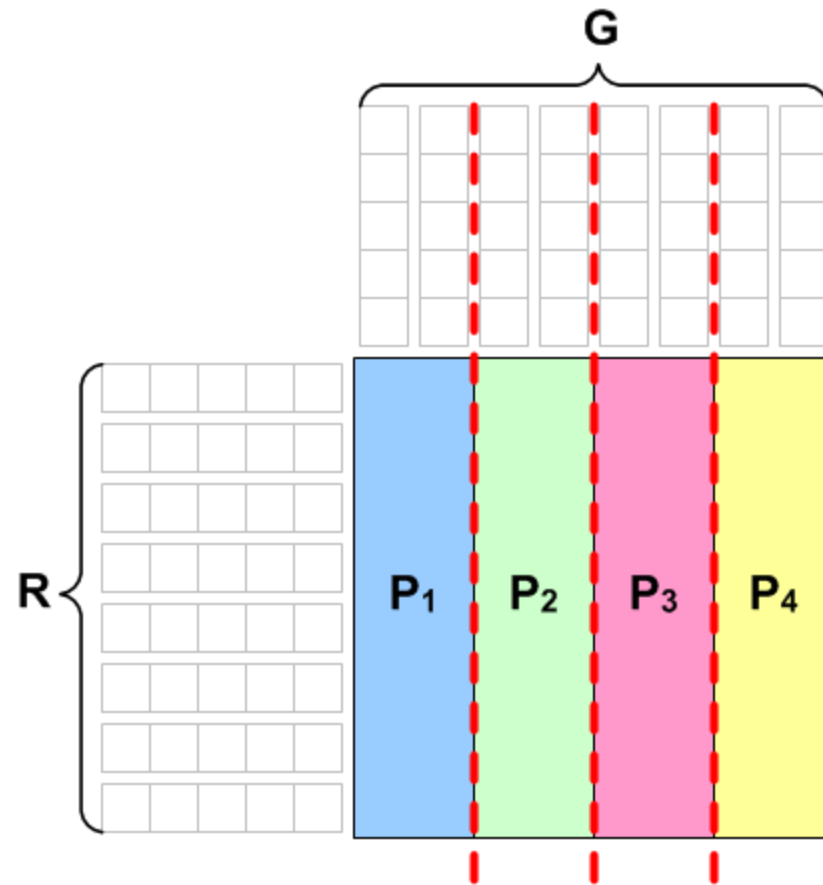
- Partition reads into N equal parts.
- Useful when R is large and G is small.
- Memory requirement does not scale



$$c_g G + (c_r + c_c G) \frac{R}{N}$$

Partitioning Genome Only (PGO)

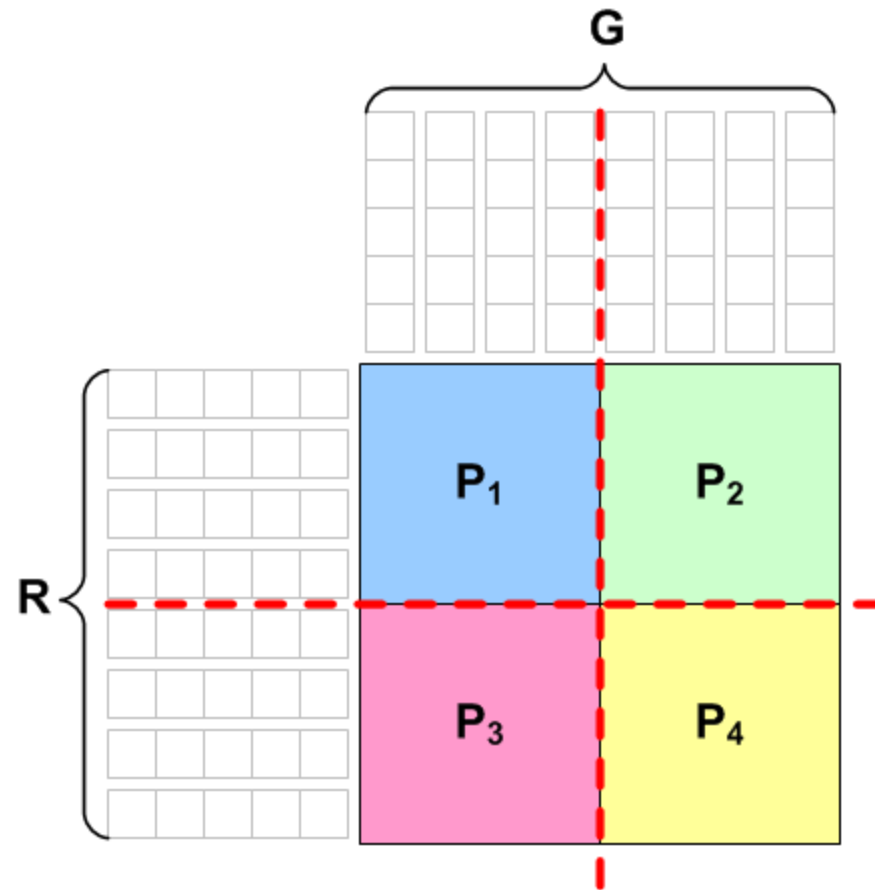
- Partition genome into N equal parts
- Useful when G is large and R is small.
- Memory requirement scales perfectly



$$c_g \frac{G}{N} + (c_r + c_c \frac{G}{N}) R$$

Partition Reads and Genome (PRG)

- A generalization of PRO and PGO
- Nodes are arranged in $N=N_1 \times N_2$ mesh
- Useful unless $G \gg R$ or $G \ll R$
- Memory scales worse than PGO, but better than PRO

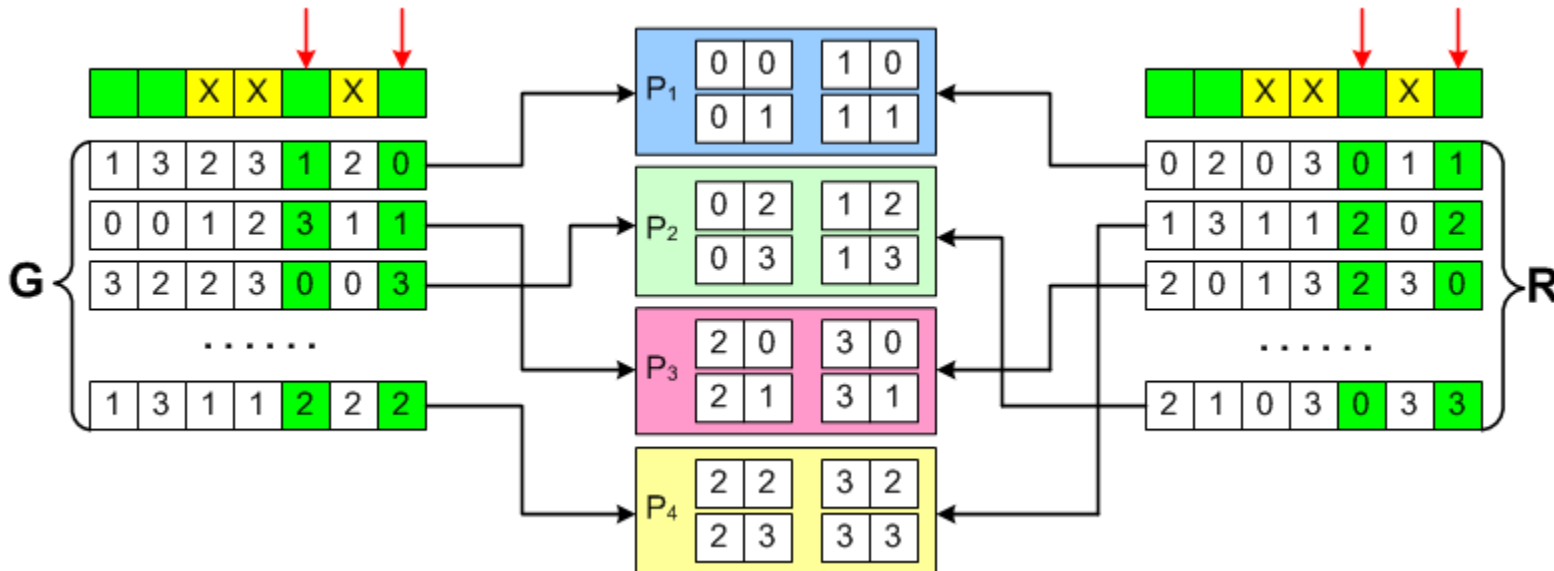


$$c_g \frac{G}{N_1} + \left(c_r + c_c \frac{G}{N_1} \right) \frac{R}{N_2}$$

Suffix Based Assignment (SBA)

- Need to avoid processing entire genome (PRO) or all reads (PGO)
- Assign a set of suffixes of length s to each node
 - 4^s suffixes for a given s
- Each node only processes reads and genome sequences that end with assigned suffixes

- Consider suffixes at the last s care positions

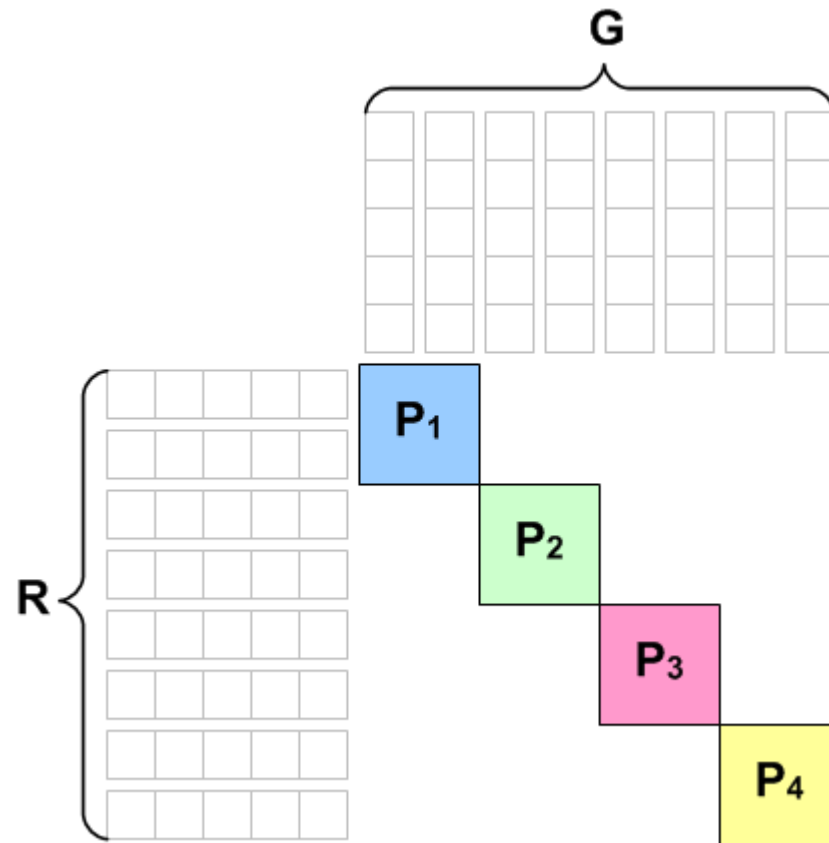


- The number of reads (genome sequences) ending in sfxA is not necessarily the same as that ending in sfxB
- Count the occurrences of each color/base in a sample of reference genome
- Estimate the load for each suffix based on occurrence frequency of colors/bases in the suffix
 - Independent of pattern being considered
- Balance the load using **bin packing**
- Balance can be improved using larger s but this increases the number of suffixes, hence comparison cost

- c_{gs} : Time to compare a genome sequence against assigned suffixes
- c_{rs} : Time to compare a read against assigned suffixes
- Assigned genome sequences are not consecutive
 - Sliding window is no longer efficient
 - c_g' : Time to compute index/hash from scratch

$$c_g' \frac{G}{N} + c_{gs}G + \left(c_r + c_c \frac{G}{N}\right) \frac{R}{N} + c_{rs}R$$

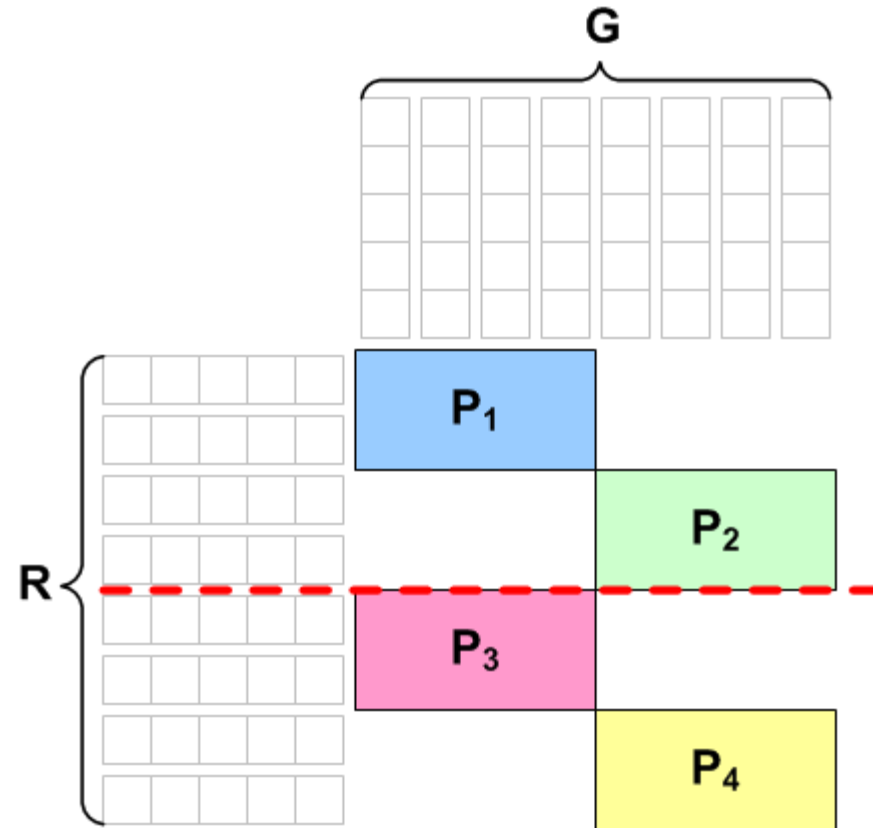
- Useful for medium values of N
 - Under perfect balance G and R are partitioned equally
 - Limited scalability due to c_{gs} and c_{rs} terms
- Memory requirement scales well



$$c'_g \frac{G}{N} + c_{gs} G + \left(c_r + c_c \frac{G}{N} \right) \frac{R}{N} + c_{rs} R$$

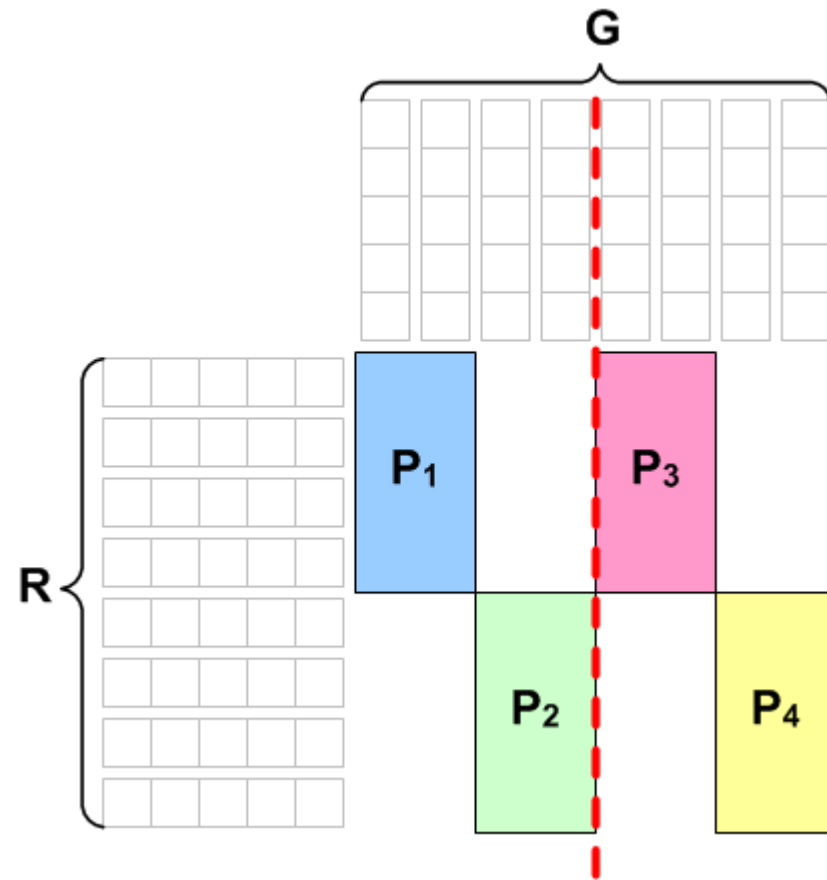
SBA after Partitioning Reads (SPR)

- Form N_2 node groups and assign R/N_2 reads to each.
- Apply SBA on each group
- Takes advantage of SBA when R is large



$$c'_g \frac{G}{N_1} + c_{gs} G + \left(c_r + c_c \frac{G}{N_1} \right) \frac{R}{N} + c_{rs} \frac{R}{N_2}$$

- Form N_1 node groups and assign G/N_1 sequences to each.
- Apply SBA on each group
- Takes advantage of SBA when G is large

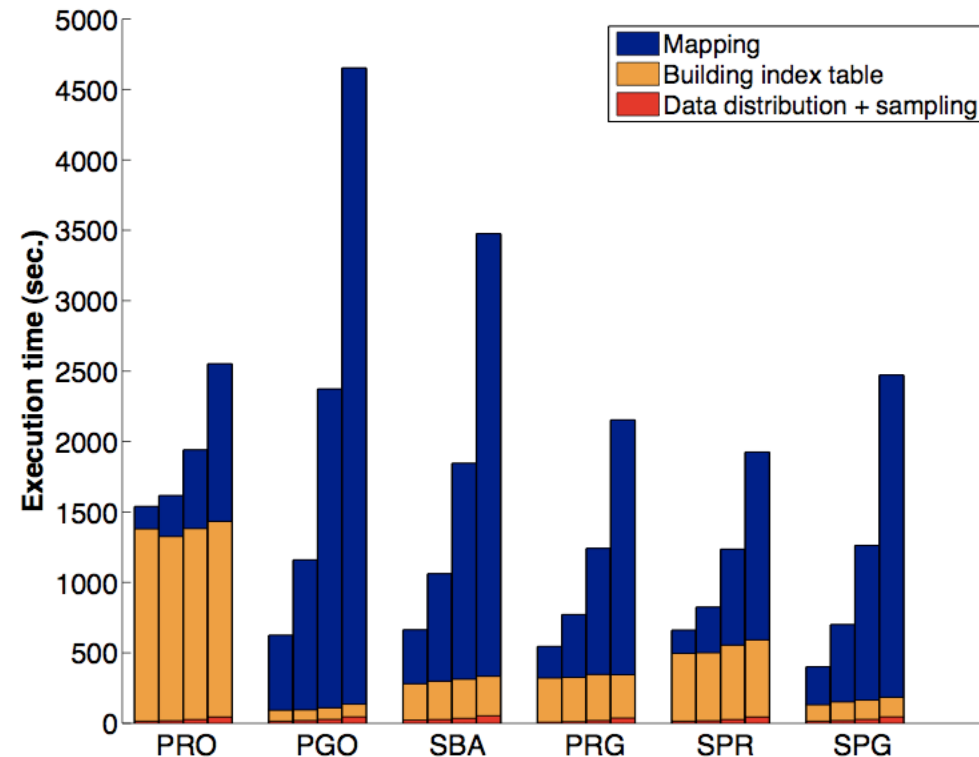


$$c'_g \frac{G}{N} + c_{gs} \frac{G}{N_1} + \left(c_r + c_c \frac{G}{N} \right) \frac{R}{N_2} + c_{rs} R$$

- Our implementation is based on *MapReads*, a part of SOLiD System Color Space Mapping Tool
 - Implemented in C using MPI
 - Used default covers with allowing up to 2 mismatches
- Experiments on 64-node dual 2.4GHz Opteron cluster with 8GB memory
- Nodes are interconnected via Infiniband MVAPICH v0.9.8
- Reads from a single run of SOLiD system
- Human Genome Build 36.1 (<http://genome.uscs.edu>)
- $N_1 = N_2 = \sqrt{N}$

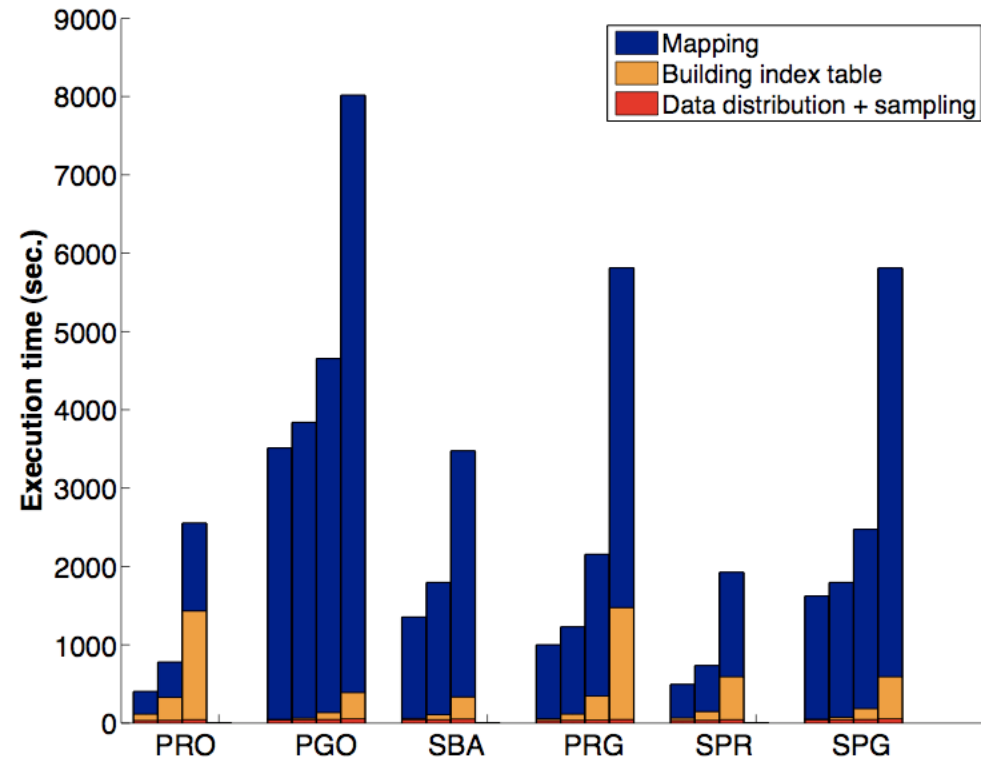
Varying Number of Reads

- G: 800M, R: (16M, 32M, 64M, 130M), N:16
- Partitioning reads helps reducing matching time



- G: (50M, 200M, 800M, 3080M), R: 130M, N:16
- Partitioning genome helps reducing hashing time

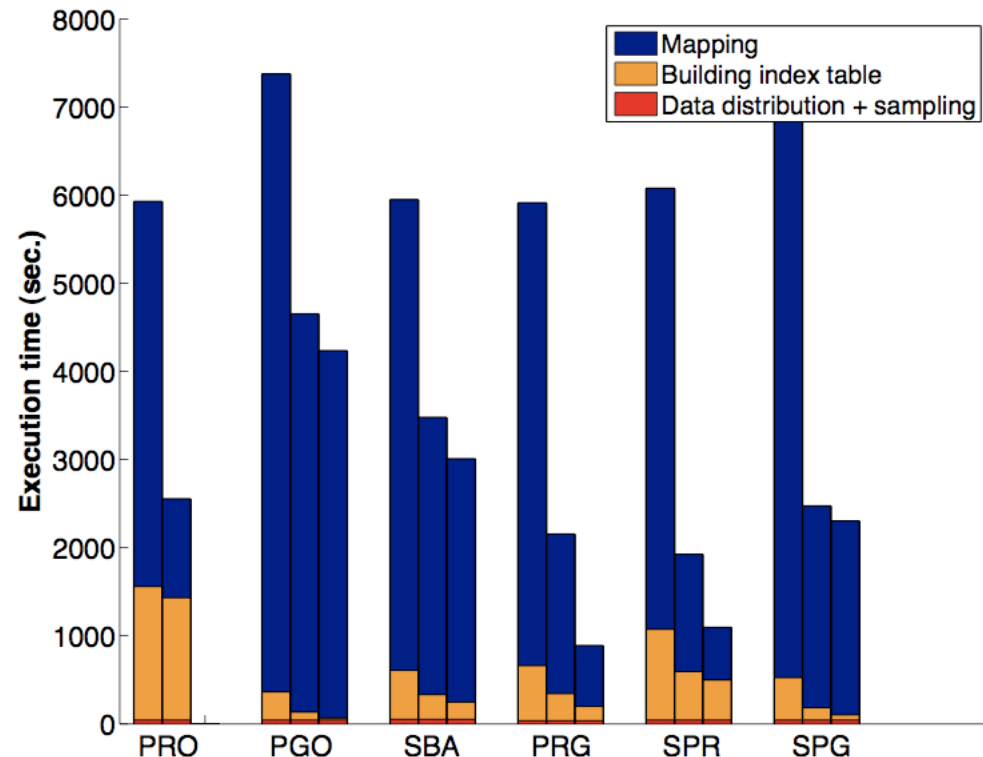
- Memory problems with PRO and SPR when G=3080M (SBA failed unexpectedly)



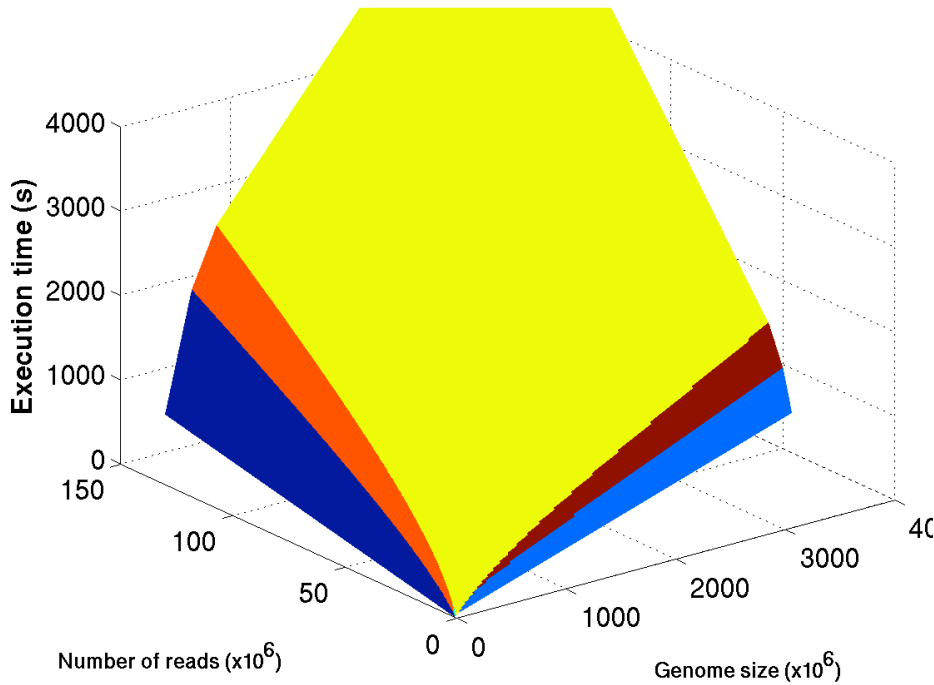
Varying Number of Nodes

- G: 800M, R: 130M, N: (4, 16, 64)

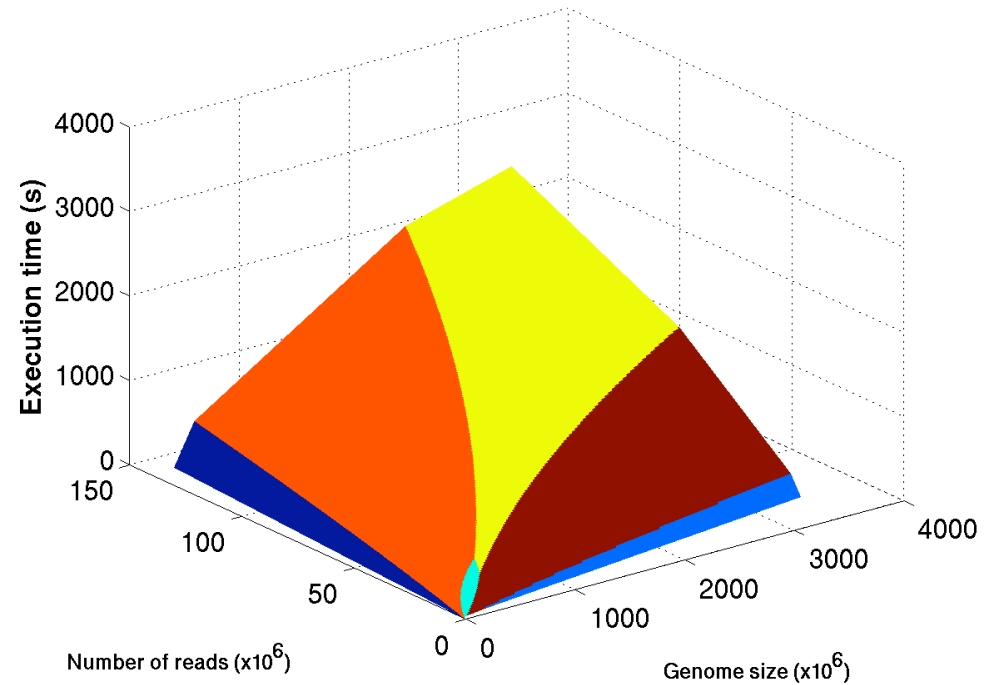
- Up to 22x speedup:
From a day to an hour!
- PRO failed unexpectedly



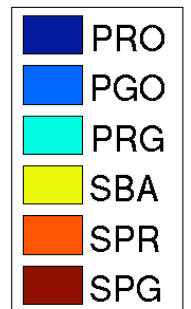
Runtime Estimates for Various Configurations



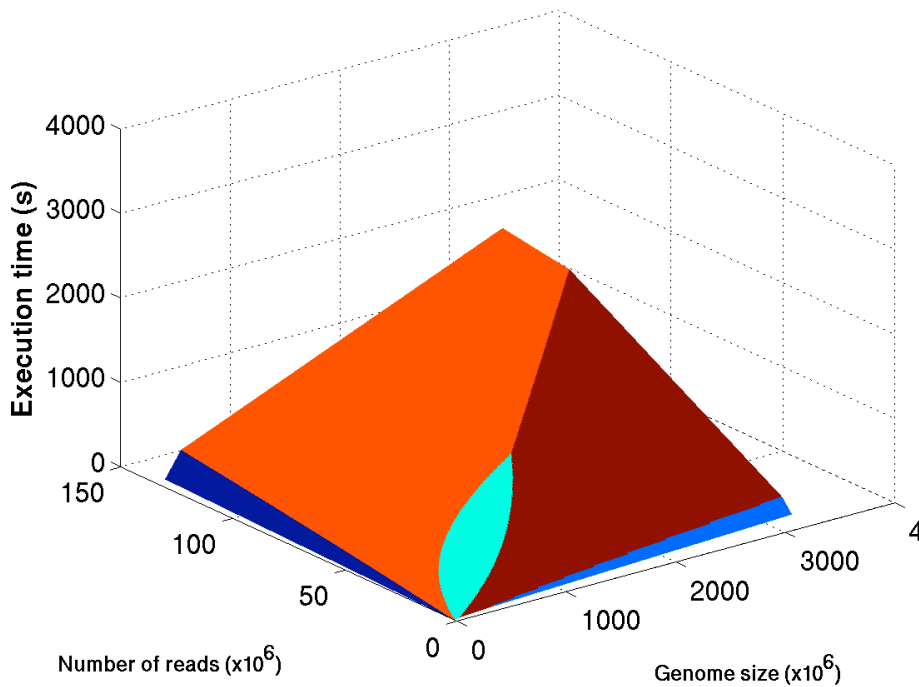
4 Processors



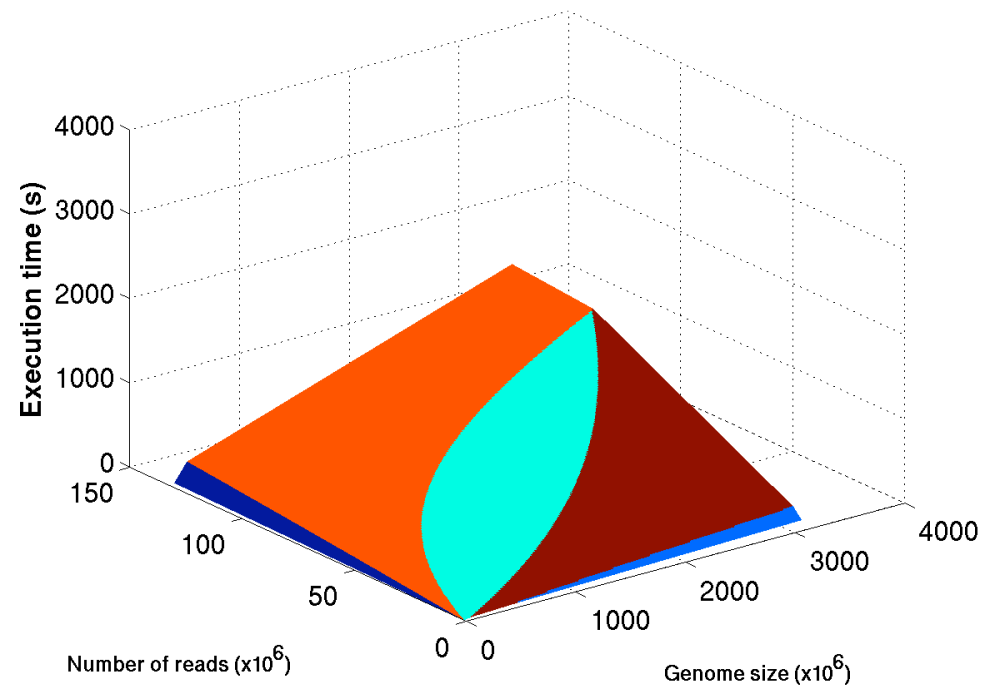
16 Processors



Runtime Estimates for Various Configurations



36 Processors



64 Processors



Coefficient	Value
c_g	1.58×10^{-6}
c'_g	1.46×10^{-6}
c_{gs}	0.22×10^{-6}
c_r	25.39×10^{-6}
c_c	0.13×10^{-12}
c_{rs}	10.15×10^{-6}

genome reads nodes	cost	PRO	PGO	SBA	PRG	SPR	SPG	speedup
800M	est.	1394	587	445	520	562	407	8.7
16M	avg.	1387	574	466	518	606	370	
16	act.	1523	608	640	537	645	385	
800M	est.	2300	4206	1829	1968	1214	2469	9.9
130M	avg.	2349	4276	2106	2074	1712	2285	
16	act.	2507	4606	3421	2116	1879	2426	
50M	est.	337	3356	1545	896	578	2165	12.5
130M	avg.	360	3418	1110	919	398	1267	
16	act.	367	3468	1310	962	458	1578	
800M	est.	1525	3526	1571	777	567	1798	17.8
130M	avg.	N/A	3464	1274	768	782	1781	
64	act.	N/A	4186	2952	850	1049	2256	

- Imbalance problem with SBA related methods

- Proposed 6 different parallelization methods for short sequence mapping
- Extensively analyzed performance of each method wrt. genome size, number of reads and number of nodes
 - Described theoretical cost models
 - Evaluated performance experimentally
- Proposed a prediction function to select the best method for a given scenario
- Achieved fairly good speedup that allows reducing the mapping time from a day to an hour.

- Fixing and cleanup
 - Investigate causes of imbalance for SBA related methods
 - Identify reasons specific to MapReads
 - Investigate additional factors in the cost functions
 - Improve prediction method to allow tolerance to potential imbalances
 - Implement the most general model that encompasses all parallelization methods
 - SBA after partitioning both reads and the genome
- Compute best values for N_1 and N_2 for given G, R and N
 - including $N_1 \times N_2 < N$
 - also a scheduling problem for multi-user server deployment
- Can we improve load balance?
- How to choose the tolerance for Cover Design?
- Can we improve the patterns of optimum Cover Designs?
- Next Problem: **Genome Assembly**

- More Info:
 - <http://bmi.osu.edu/~umit>
 - umit@bmi.osu.edu