RadixZip: Linear Time Compression of Token Streams

Binh Vo <binh@google.com>
Gurmeet Singh Manku <manku@google.com>
Google Inc., USA

Data of interest

- Collections of records:
  - Databases.
  - Logs (query or ad-clicks at Google).
  - Tables (telephone records at AT&T).
- Transposing into collections of columns.
  - Faster lookup of specific attributes.
  - Improved compression.

Context sorting compressors

- BZip - 1994 (Burrows, Wheeler, Seward).
  - General purpose compression.
  - Based on the BWT (suffix sorting).
- Vczip - 2004 (Vo and Vo).
  - Fixed width table compression.
  - Based on column dependency (predictor sorting).
- Common theme: sort data by some context.
  - A context is any string which helps 'predict' target.
  - Similar to sorting the target if prediction is accurate.
  - But reversible!

BWT: Suffixes as a context

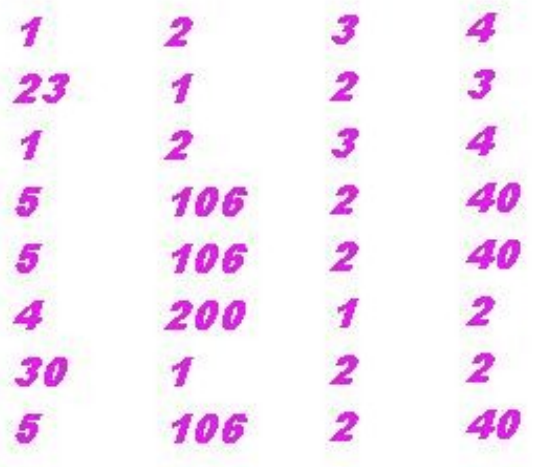
t h a t i s t h a t

h	atisthat
h	at
t	hat
t	hatisthat
t	isthat
i	sthat
a	tisthat
a	t
s	that
t	

- Transformed data is more compressible.
  - Bzip = BWT + Move-to-Front + Run-Length + Huffman

Column-specific properties

- Boundary awareness:
  - Byte indices.
  - Intra-token contexts.
- Multi-column context:
  - Dependency.
  - E.g. a user with a fixed IP and browser.



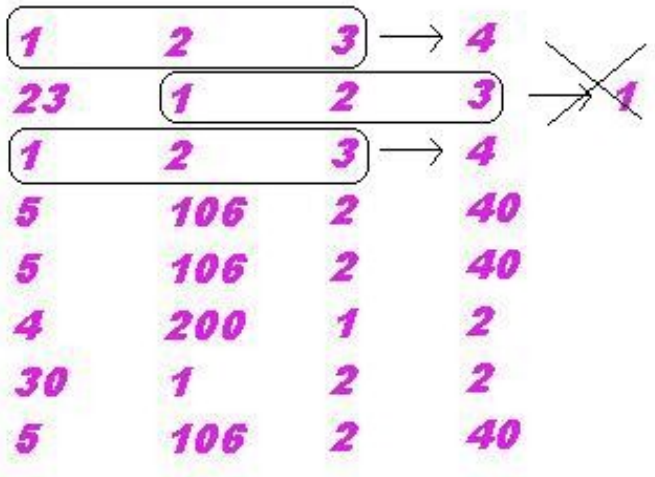
Token-specific redundancy

- Boundary awareness:
  - Byte indices.
  - Intra-token contexts.
- Multi-column context:
  - Dependency.
  - E.g. a user with a fixed IP and browser.

1	2	3	4
23	1	2	3
1	2	3	4
5	106	2	40
5	106	2	40
4	200	1	2
30	1	2	2
5	106	2	40

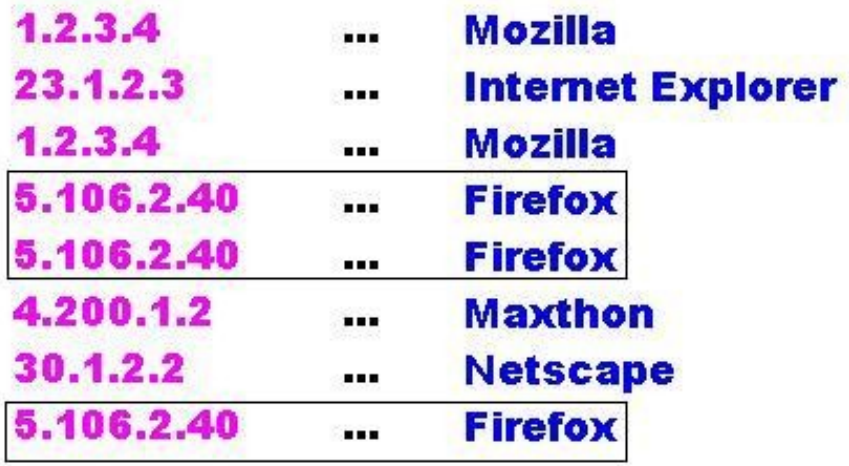
Token-specific redundancy

- Boundary awareness:
  - Byte indices.
  - Intra-token contexts.
- Multi-column context:
  - Dependency.
  - E.g. a user with a fixed IP and browser.



Token-specific redundancy

- Boundary awareness:
  - Byte indices.
  - Intra-token contexts.
- Multi-column context:
  - Dependency.
  - E.g. a user with a fixed IP and browser.



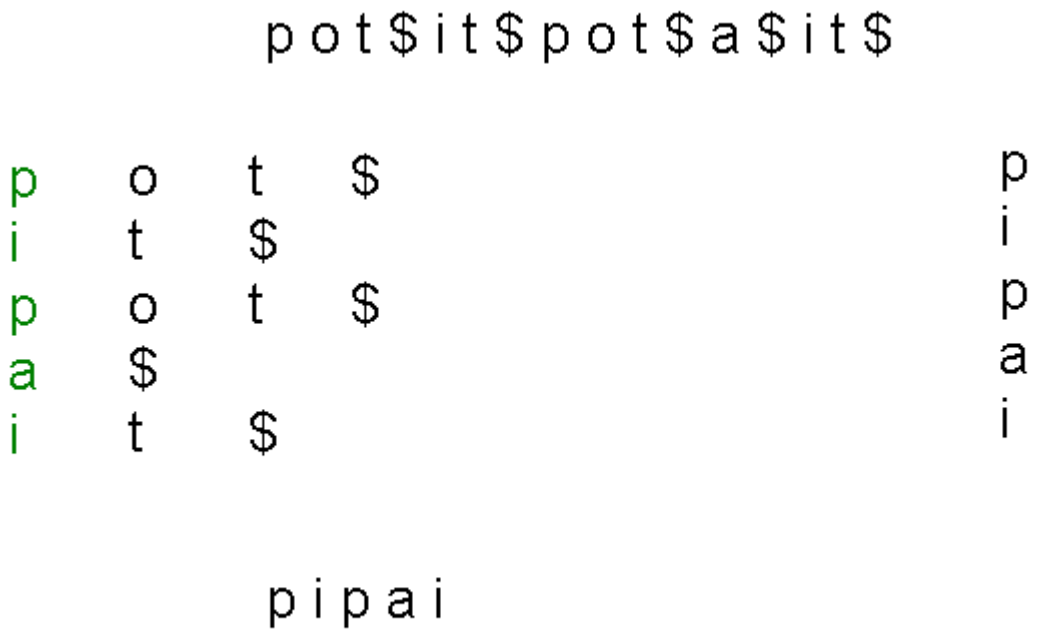
RadixZipTransform

p o t $ i t $ p o t $ a $ i t $

p	o	t	$
i	t	$	
p	o	t	$
a	$		
i	t	$	

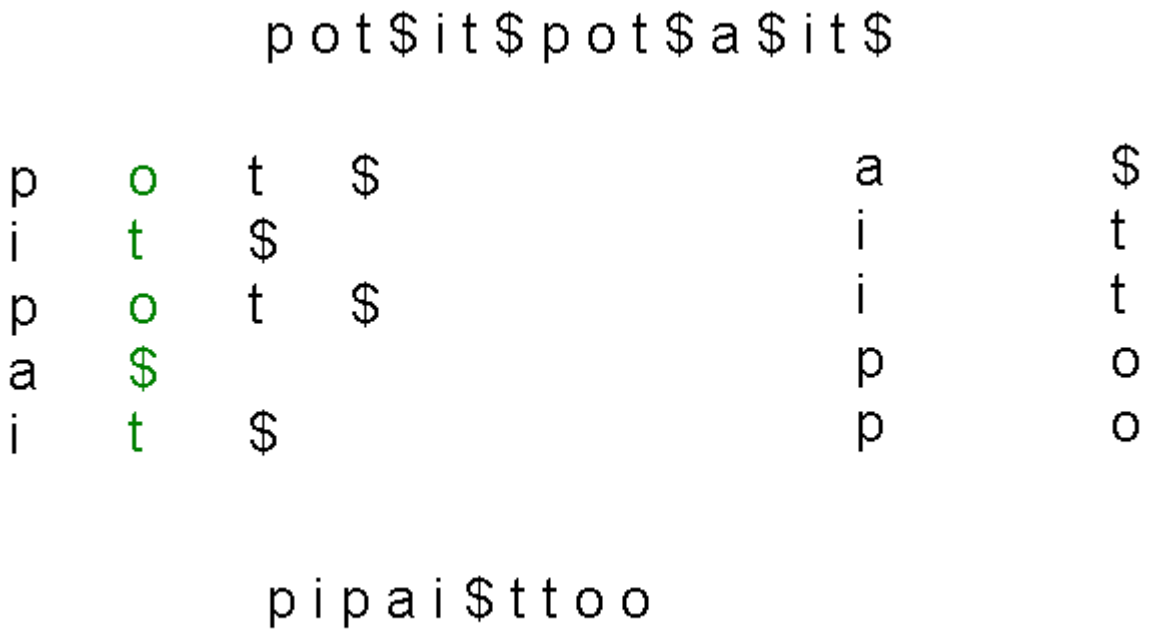
- For each col *i:*
  - Sort by token prefixes formed from earlier columns.
  - Append reordered col *i* to output.

RadixZipTransform



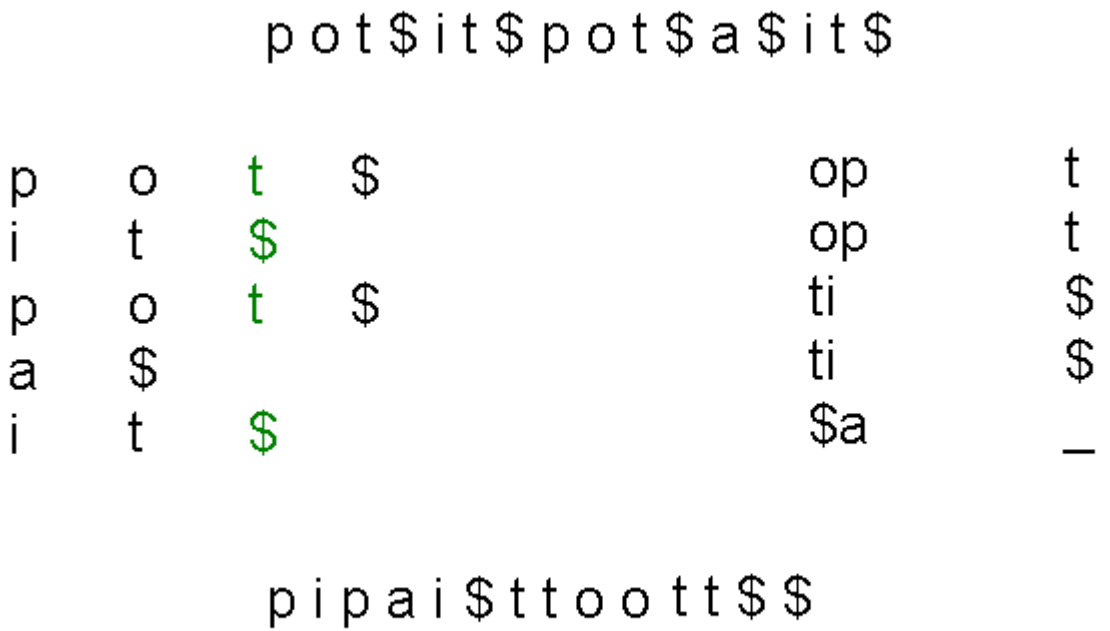
- For each col *i:*
  - Sort by token prefixes formed from earlier columns.
  - Append reordered col *i* to output.

RadixZipTransform



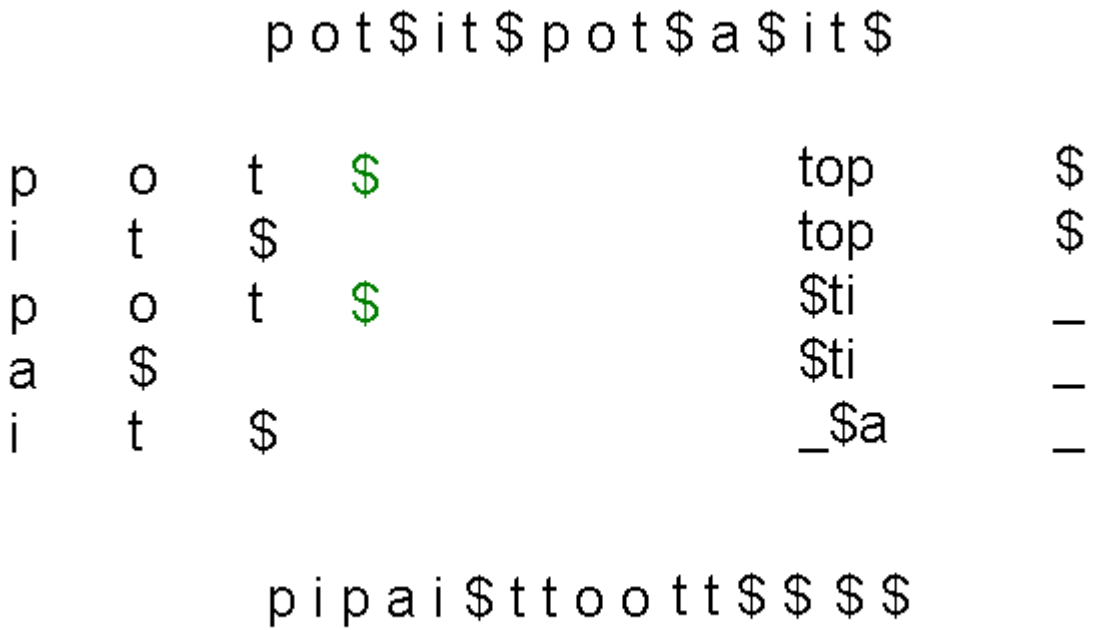
- For each col *i:*
  - Sort by token prefixes formed from earlier columns.
  - Append reordered col *i* to output.

RadixZipTransform



- For each col *i:*
  - Sort by token prefixes formed from earlier columns.
  - Append reordered col *i* to output.

RadixZipTransform



- For each col *i:*
  - Sort by token prefixes formed from earlier columns.
  - Append reordered col *i* to output.

Linear Time

p	o	t	$
i	t	$	
p	o	t	$
a	$		
i	t	$	

p i p a i $ t t o o t t $ $ $ $

- Perform a Radix sort.
- Append one column before each iteration.

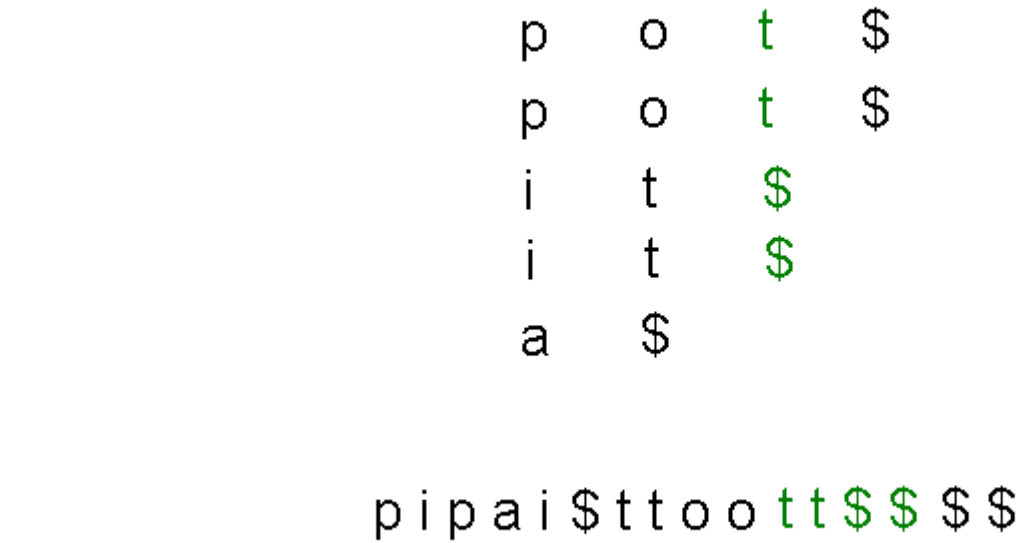
Linear Time

a	$		
i	t	$	
i	t	$	
p	o	t	$
p	o	t	$

p i p a i $ t t o o t t $ $ $ $

- Perform a Radix sort.
- Append one column before each iteration.

Linear Time



- Perform a Radix sort.
- Append one column before each iteration.

Linear Time

p	o	t	$
p	o	t	$
i	t	$	
i	t	$	
a	$		

p i p a i $ t t o o t t $ $ $ $

- Perform a Radix sort.
- Append one column before each iteration.

Compression benefits

o n e $
t w o $
o n e $
o n e $
t h r e e $
t w o $
o n e $
t h r e e $
o n e $

RadixZipSort: otoottotonnnnnwhwhrreeeeeoo$$$$$$$eeee$$

BWT(prefix): o$$$$$$$eerreeeeen$$nnnneewwhhootottooo

- Preserves byte columns.
- Context sorted, but limited to token boundaries.
- Transformed data is more compressible:
  - RadixZip = RadixZipTransform + MTF + RLE + Huffman

Performance

- Linear time complexity.
- Memory properties:
  - Requires 8 bytes per token.
  - Cache-friendly.
- Comparison to BWT:
  - Faster than currently known BWT implementations.
  - Similarly, using less memory.
  - RadixZip is simple to implement, robust code.

Inter-column dependency

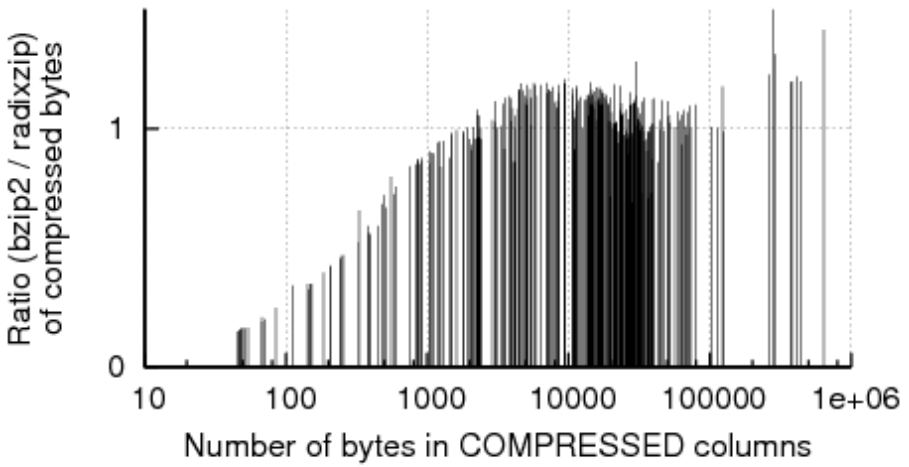
Binh	1	94040	94040
Joe	6	02136	94040
Binh	2	94040	94040
Google →	4	94043 ⟶	94043
Binh	3	94040	94043
Google	5	94043	02136
Joe	7	02136	02136

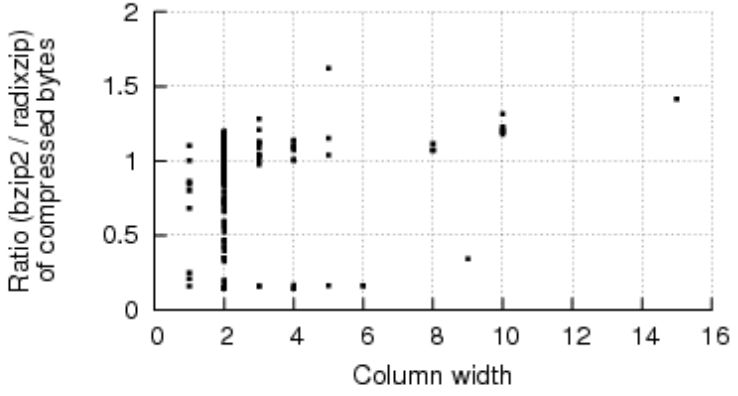
No input: 9099990224444411000004444433660303
Input: 9999900224444411000004444433660033

- Passing permutations equivalent to presorting.
- Passed permutations continue to propagate.

RadixZip vs Bzip2 (census data)

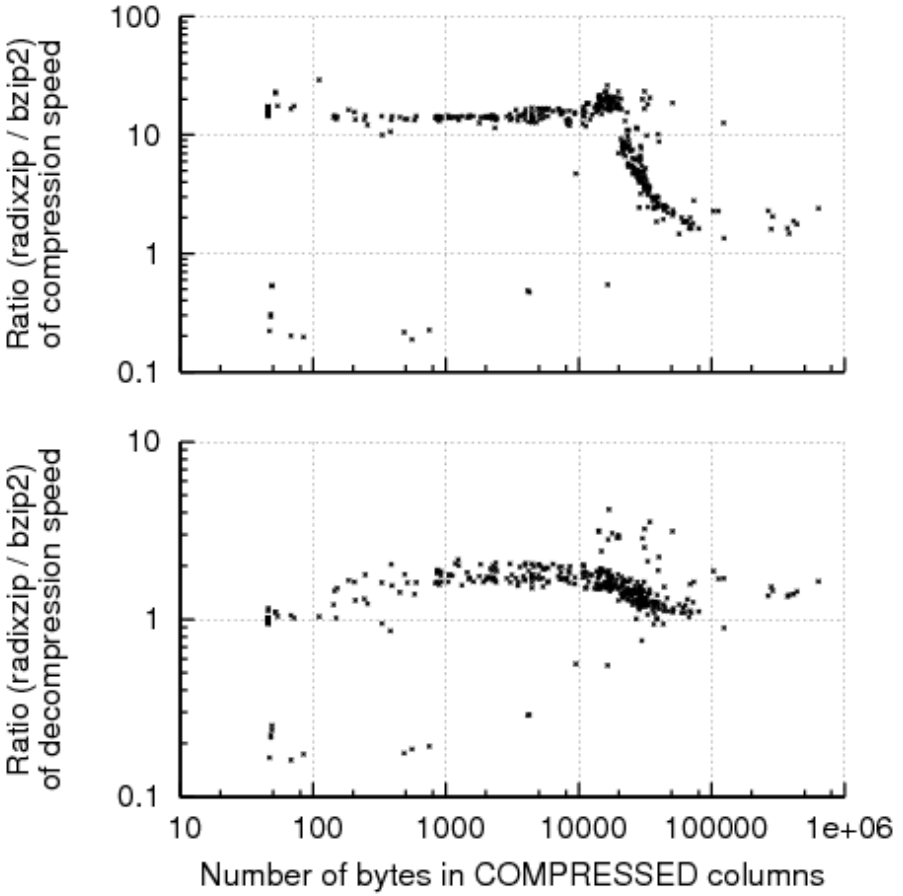
- US population survey.
  - Fixed-width fields.
  - Divided by field.
- RadixZip outperforms on larger columns.
- Loss on smaller ones,
  - Likely due to needing more byte-columns to 'ramp up'.
- About 15% total gain.





RadixZip vs Bzip2 (census data)

- Compression speed improves:
  - Especially on highly compressible streams,
  - Since Bzip2's alg is worst-case quadratic.
- Decompression speed improves.
- Most outliers are on very small streams.



Dependency results

- Hand-picked dependencies from census data.
- Use of a predictor can reduce compressed size to ~0.
- High dependency indicates little to no new information.

bzip2	RadixZip	predictor	% reduction
16003	16357	32	99.8 %
13741	13348	28	99.8 %
21559	20555	63	99.7 %
16156	16310	45	99.7 %
14518	19751	86	99.6 %
12076	12398	52	99.6 %
10078	10664	44	99.6 %
26213	25187	120	99.5 %
24948	23392	124	99.5 %
12084	12416	62	99.5 %
29503	27586	153	99.4 %
11650	12091	73	99.4 %
11992	12346	99	99.2 %
11681	12109	110	99.1 %
10681	10185	95	99.1 %

Conclusion

- RadixZipTransform - a linear time transform.
- Improvement in both performance and compression for token streams over general purpose compressors.
- Efficient exploitation of stream correlation.