Leftmost Longest Regular Expression Matching in Reconfigurable Logic

Kubilay Atasu
IBM Research - Zurich

Presented at FPT 2015
Queenstown, NZ

Outline

- **Introduction**
- A general architecture
- An optimized architecture
- Experiments and results

Text analytics: Extracting information from unstructured text

- distill structured data from unstructured and semi-structured text
- exploit the extracted data in your applications

For years, Microsoft Corporation CEO Bill Gates was against open source. But today he appears to have changed his mind. "We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access.“

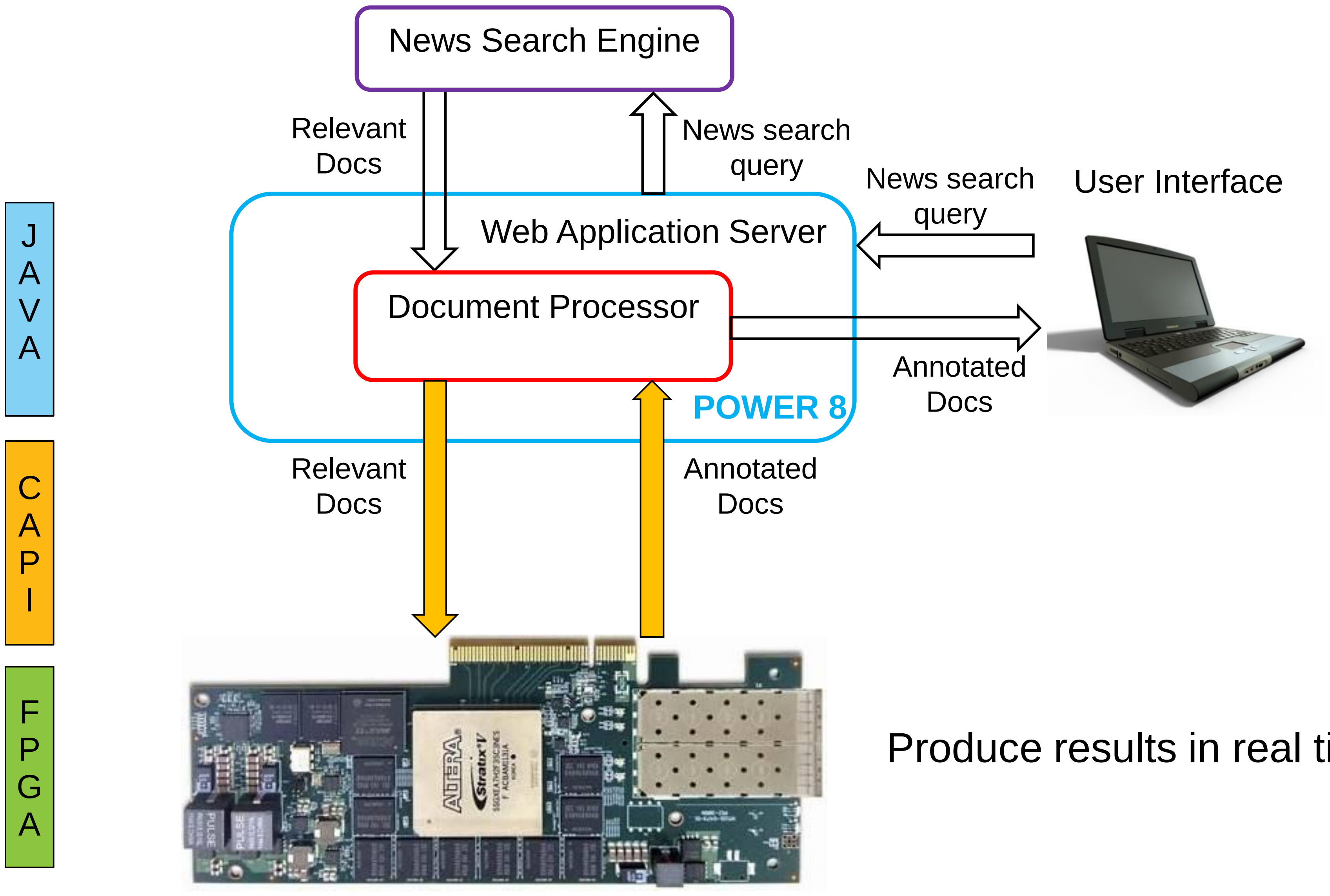
Richard Stallman, founder of the Free Software Foundation, countered saying…

Annotations

Name	Title	Organization
Bill Gates	CEO	Microsoft
Bill Veghte	VP	Microsoft
Richard Stallman	Founder	Free Soft..

(from Cohen’s IE tutorial, 2003)

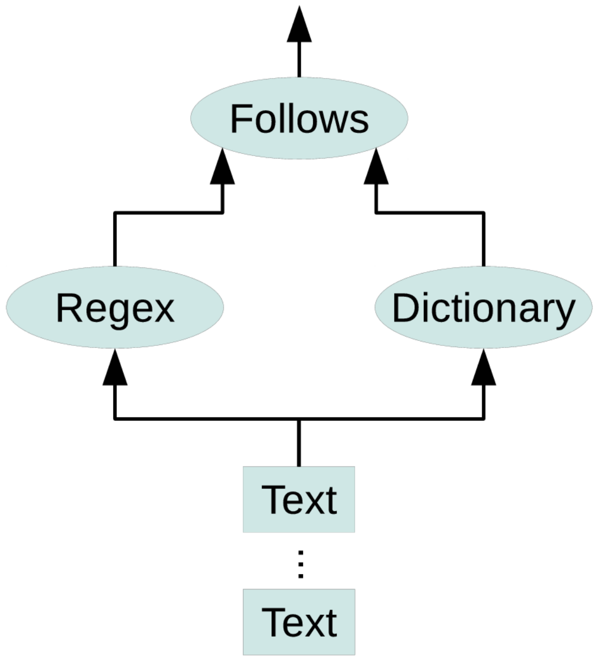
Acceleration of news search services

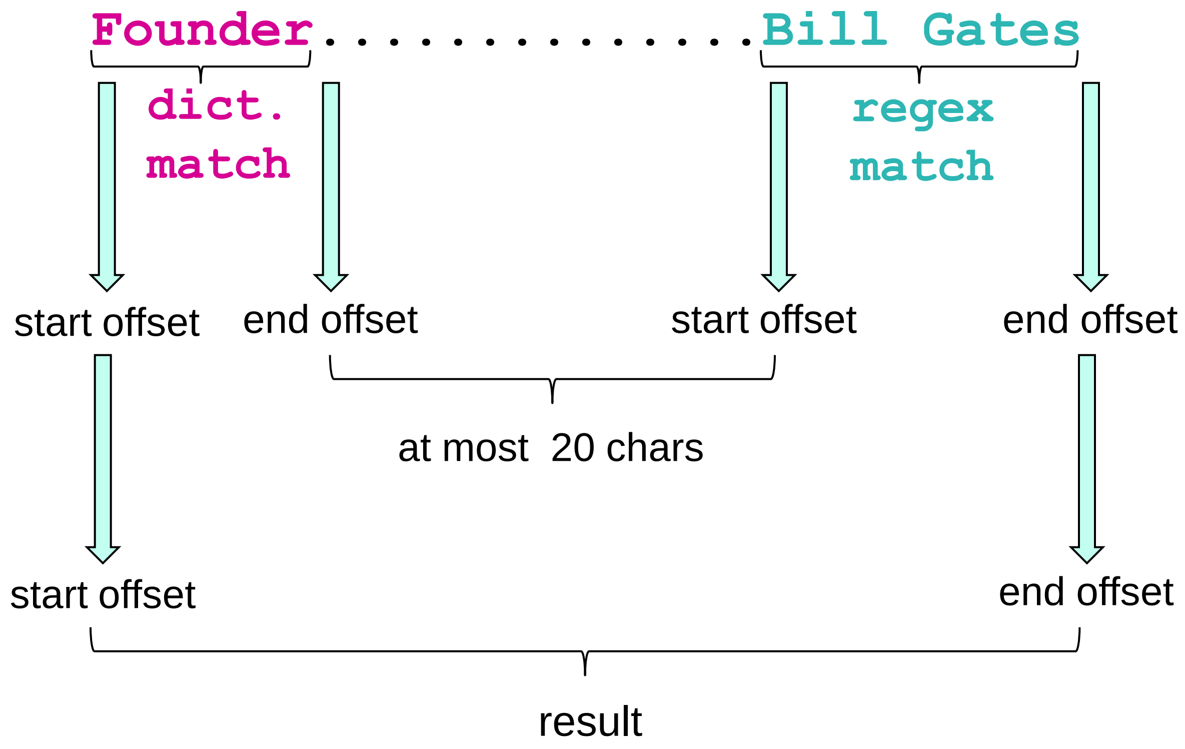


Produce results in real time!

A simple information extraction rule

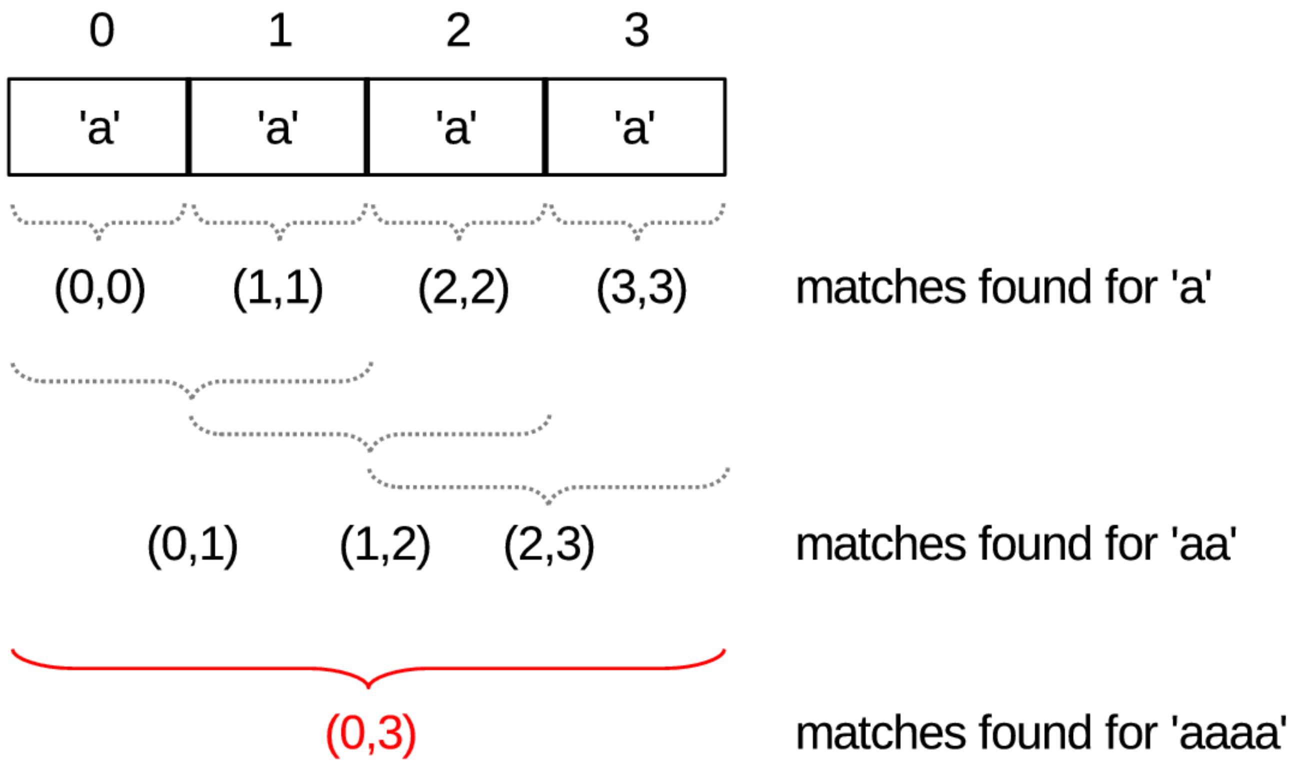
- Find the names (`regex`) that are at most 20 chars after a title (`dict.`)





Leftmost longest regular expression matching

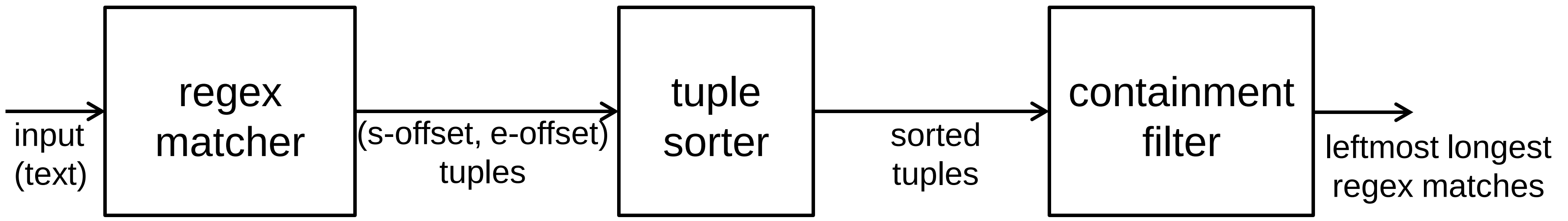
- Consider the regex (a|aa|aaaa)
- Consider the input string aaaa
- There are eight distinct regex matches
- A single leftmost longest match (in red)



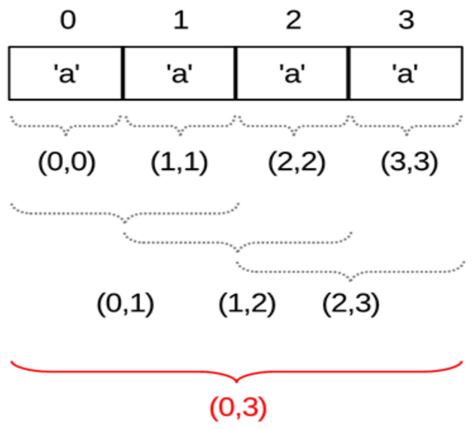
Outline

- Introduction
- **A general architecture**
- An optimized architecture
- Experiments and results

A general leftmost longest regex matching architecture



- Produce (start offset, end offset) tuples for the regex matches
- Sort the tuples in the increasing order of start offsets
  - sort in the decreasing order of end offsets if start offsets are equal
- Alternatively, sort the tuples in the decreasing order of end offsets
  - sort in the increasing order of start offsets if end offsets are equal
- Eliminate tuples contained by others using a containment filter unit



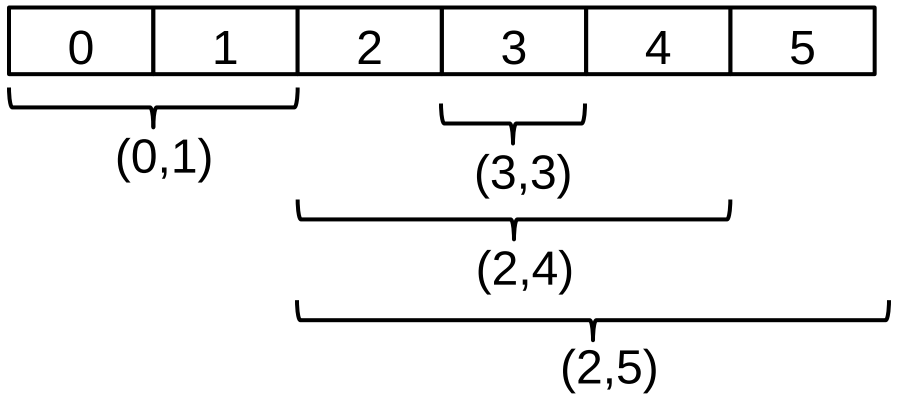
all matches: (0,0),(1,1),(2,2),(3,3),(0,1),(1,2),(2,3),(0,3)

after sorting: (0,3),(0,1),(0,0),(1,2),(1,1),(2,3),(2,2),(3,3)

after containment: (0,3)

Containment filter: Algorithm

- Needs to remember a single tuple (s0, e0) and whether it is valid or not
- When a new tuple (s1, e1) arrives:
  - if this is the first tuple then copy (s1, e1) to (s0, e0) and set the valid bit
  - else if ((s1>=s0) & (e1<=e0)) then consume (s1, e1) without producing output
  - else output (s0, e0) and copy (s1, e1) to (s0, e0)
- When input eos arrives: output (s0, e0) if the valid bit is set and clear the valid bit
  - produce output eos

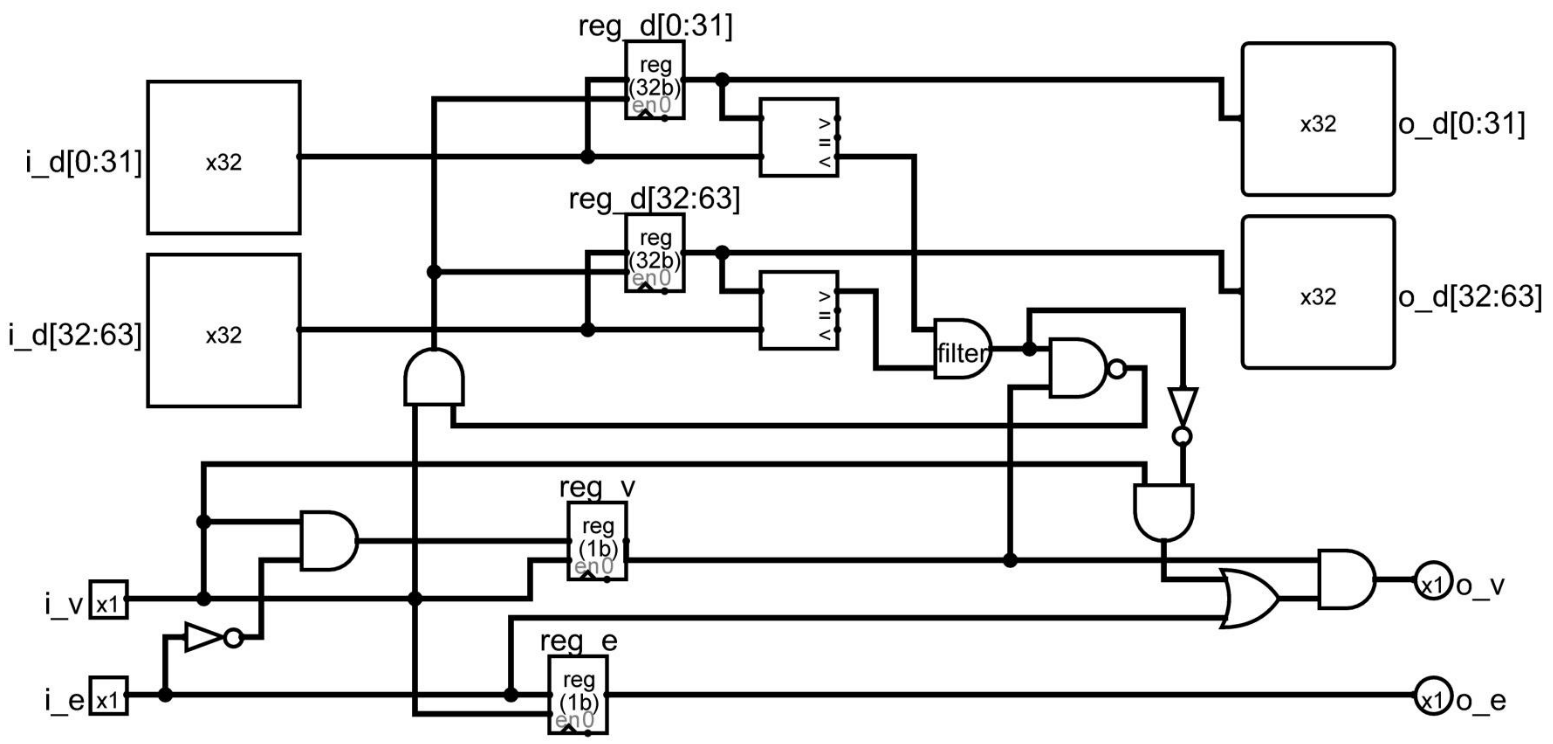


matches: (0,1), (3,3), (2,4), (2,5)

sorted: (0,1), (2,5), (2,4), (3,3)

after containment: (0,1), (2,5)

Containment filter: Design



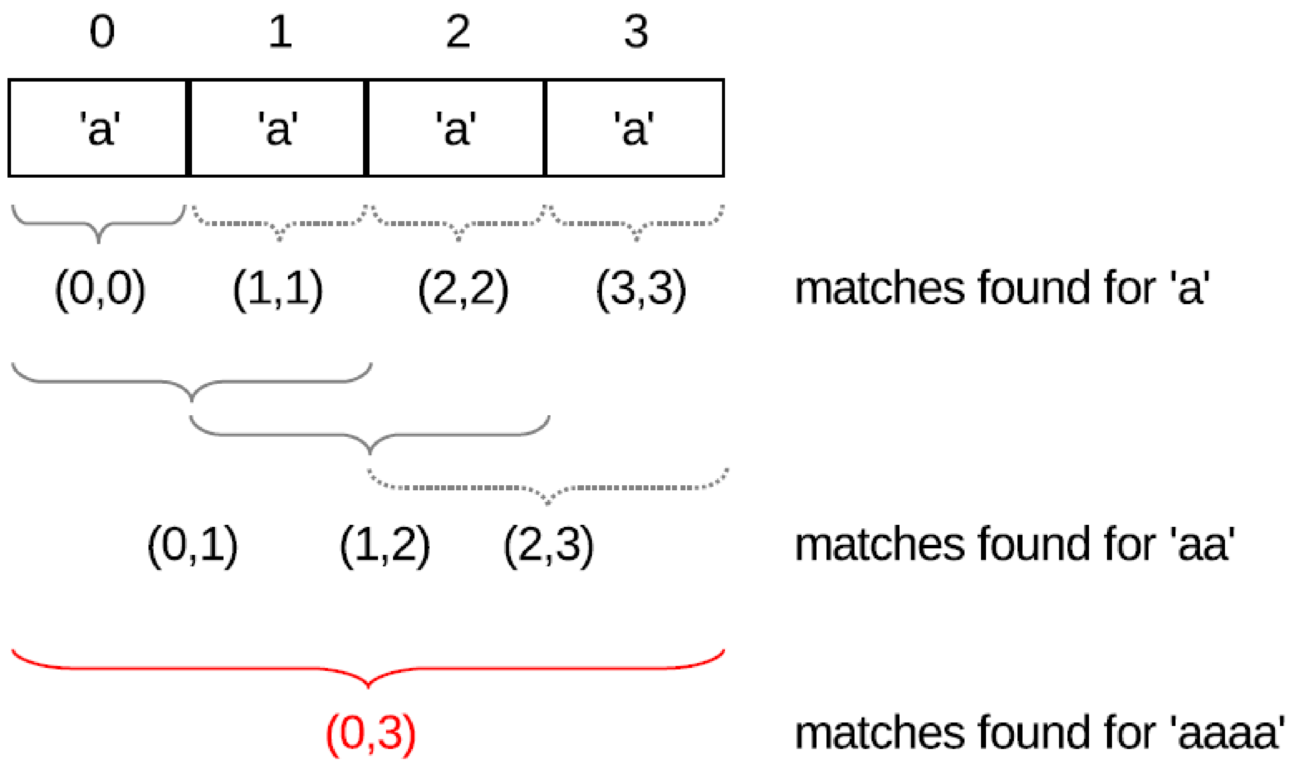
- If sorted in the increasing order of start offsets, no need to check for (s1 >= s0)
    - filter out (s1, e1) if (e1 <= e0)
- If sorted in the decreasing order of end offsets, no need to check for (e1 <= e0)
    - filter out (s1, e1) if (s1 >= s0)

Outline

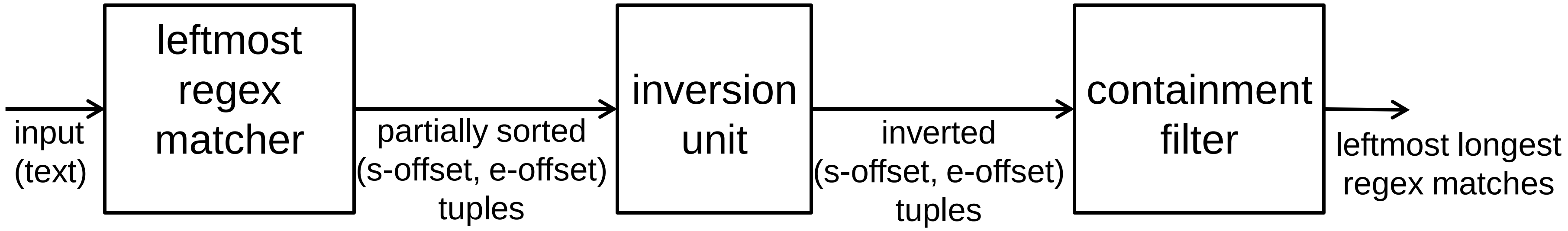
- Introduction
- A general architecture
- **An optimized architecture**
- Experiments and results

Algorithm for computing leftmost regular expression matches

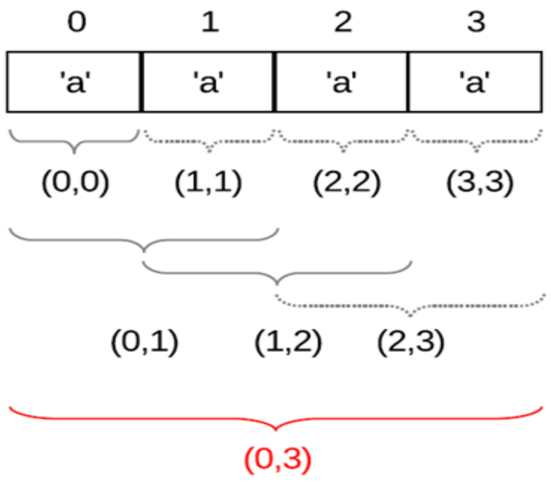
- while i in 0 to input_length-1
  - find the match with the smallest start offset that ends at offset position i
- The leftmost maches are marked using solid lines in the below example
- Prior art: Atasu et al: FPL 2013, ASAP 2014, US Patent App. 14/184,751



Computing leftmost longest matches without sorting (example 1)



- Use a leftmost regex matcher as a building block
  - produces the leftmost regex matches in the increasing order of end offsets
  - a single match with the smallest start offset can be reported per end offset
- Feed the output of the regex matcher into the containment unit in the inverse order
  - matches are now sorted in the decreasing order of end offsets
  - no two tuples can have the same end offset

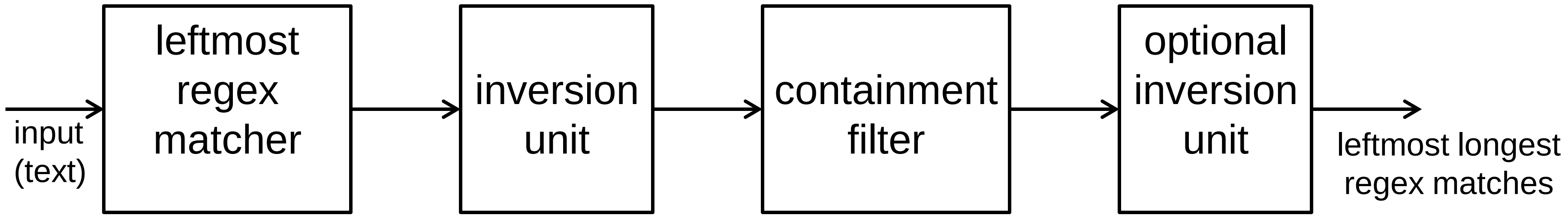


leftmost matches: (0,0),(0,1),(1,2), (0,3)

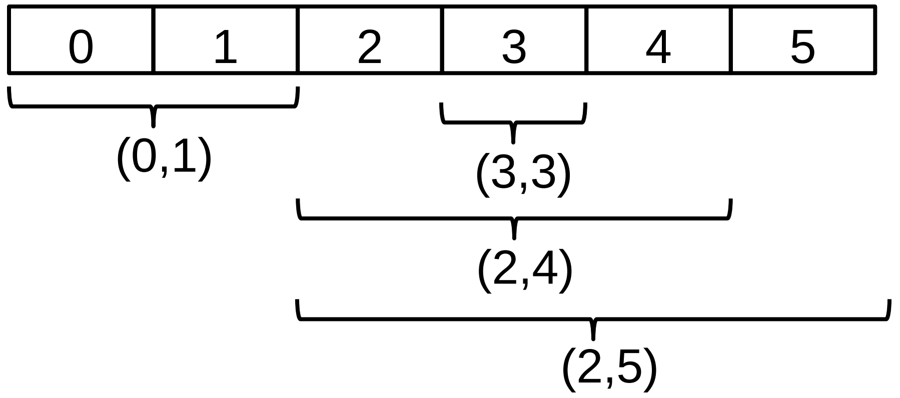
after inversion: (0,3),(1,2),(0,1),(0,0)

after containment: (0,3)

Computing leftmost longest matches without sorting (example 2)



- Use a leftmost regex matcher as a building block
  - produces the leftmost regex matches in the increasing order of end offsets
  - a single match with the smallest start offset can be reported per end offset
- Feed the output of the regex matcher into the containment unit in the inverse order
  - matches are now sorted in the decreasing order of end offsets
  - no two tuples can have the same end offset



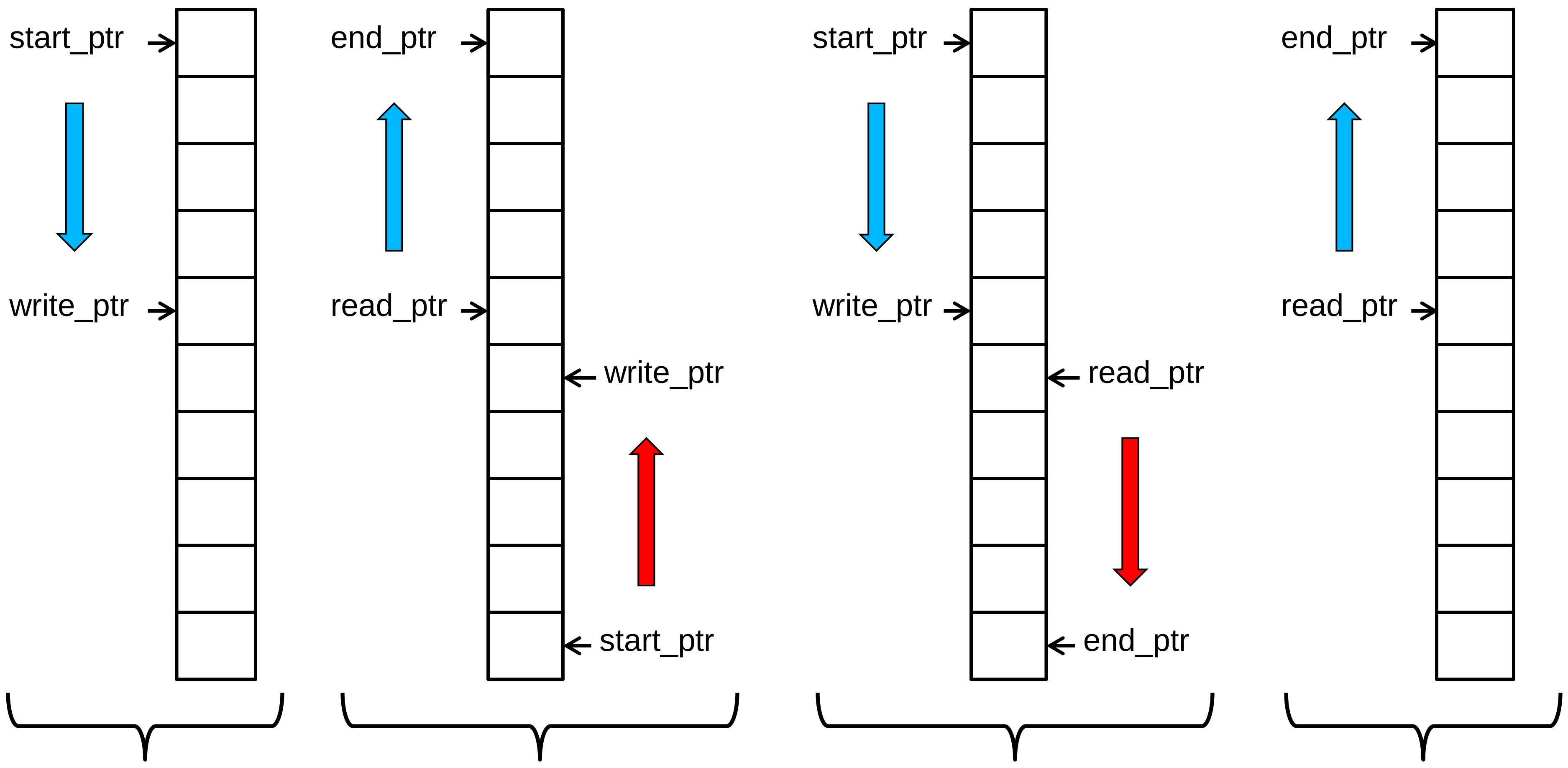
matches: (0,1), (3,3), (2,4), (2,5)

inverse: (2,5), (2,4), (3,3), (0,1)

after containment: (2,5), (0,1)

optional inversion: (0,1), (2,5)

A latency hiding inversion unit: overlapping read/write latencies



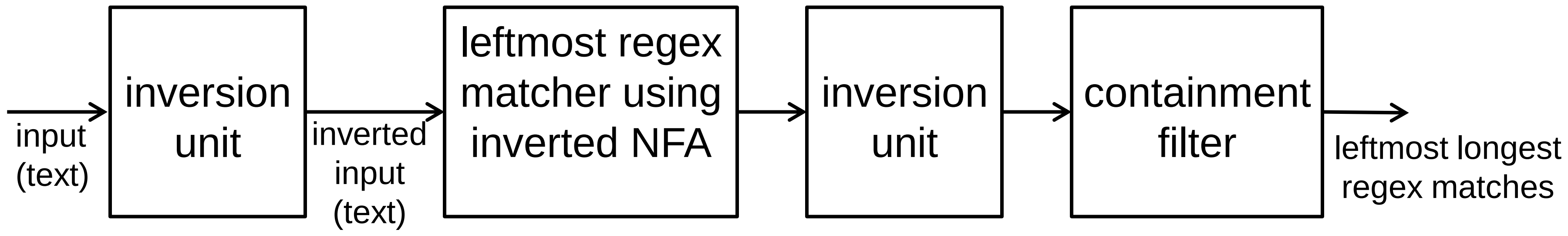
forward mode
nothing to read
write stream 0

backward mode
read stream 0
write stream 1

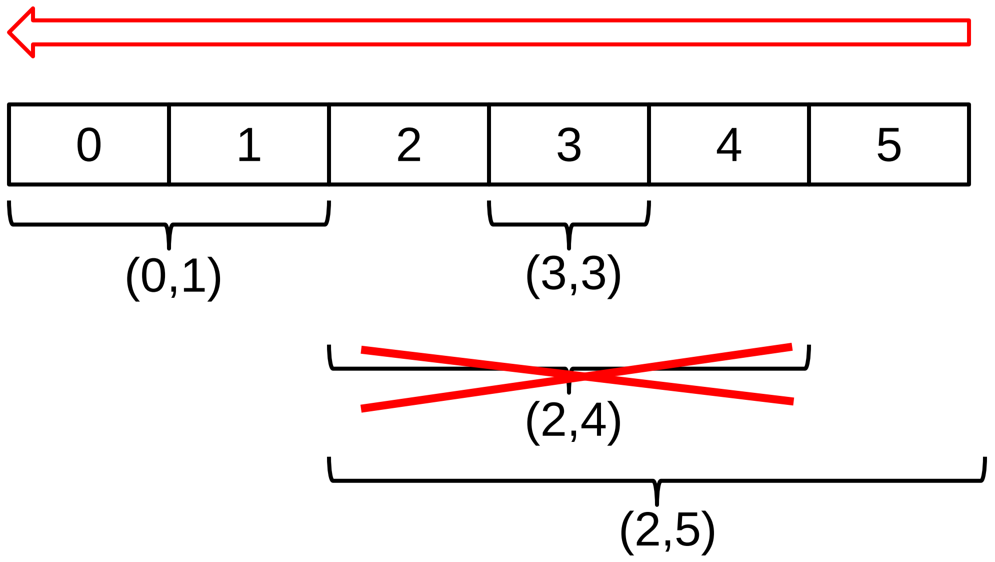
forward mode
read stream 1
write stream 2

backward mode
read stream 2
nothing to write

Computing rightmost longest regex matches



- Invert the NFA of regex, e.g., search for cba instead of abc
- Invert the input stream, i.e., search in the opposite direction
- Invert the match results (the result is sorted) and apply containment



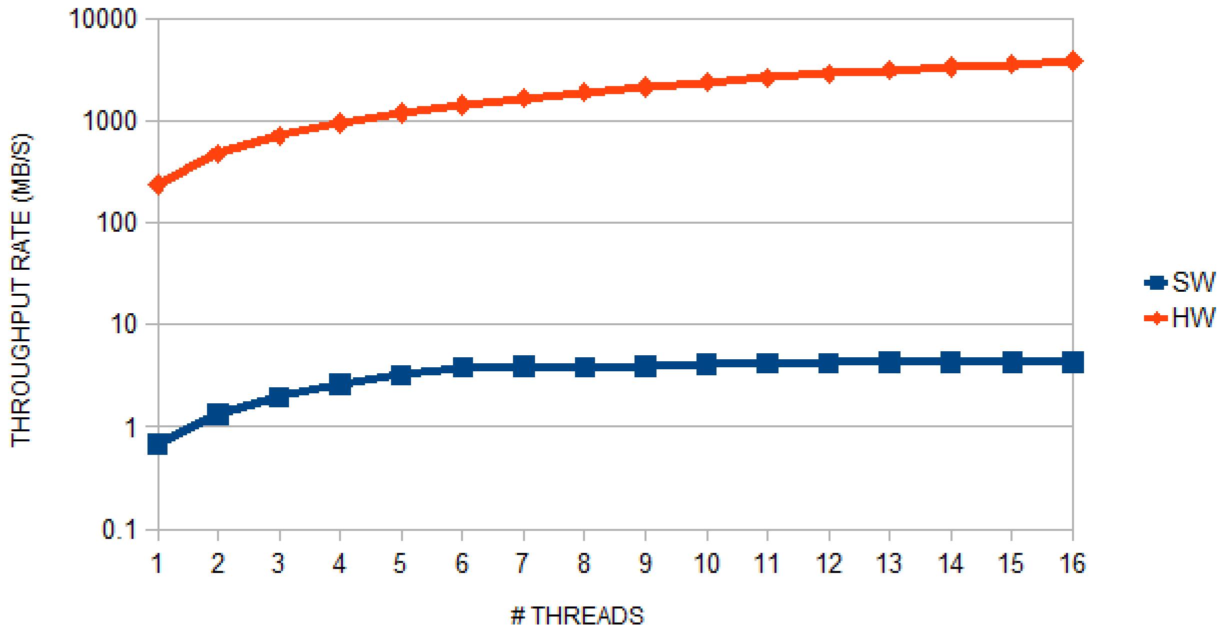
matches:	(3,3), (2,5), (0,1)
inverse:	(0,1), (2,5), (3,3)
after containment:	(0,1), (2,5)

Outline

- Introduction
- A general architecture
- An optimized architecture
- **Experiments and results**

Experiments

- HW: Altera Stratix IV GX530KH40C2, Altera Quartus II V11 tools
  - 32-bit start and end offset registers, 250 MHz target clock frequency
  - 25 text analytics regexs, 256-element deep LIFO buffers per regex
- SW: 12-core Intel ® Xeon ® E5-2630 processor, running at 2.6 GHz



- Inversion unit + Containment Unit: ~100 LUTs + ~90 regs + 2 M9K blocks
- Measured speed-up when using 4 HW threads @ ~0.95 GB/s: ~220 fold
- Estimated speed-up when using 16 HW threads @ ~3.8 GB/s: ~880 fold

Summary & Conclusions

- A baseline architecture for finding leftmost longest regex matches:
  - a regex unit that reports start and end offset positions of the matches
  - a sorter unit that sorts the match results based on start & end offset positions
  - a containment filter that eliminates the results that are not leftmost longest

- An optimized architecture for finding leftmost longest regex matches:
  - a regex unit that supports start offset reporting and leftmost matching
    - producing results in the increasing order of end offset positions
  - a LIFO unit that inverts the results computed by the regex matching unit,
    - producing results in the decreasing order of end offset positions
  - a filter (containment) unit that operates on the result of the LIFO unit
    - filtering out matches having an equal or larger start offset

- Adaptation of these architectures to compute the rightmost longest regex matches

- An FPGA implementation that achieves > 200 fold improvement in performance

Leftmost Longest Regular Expression Matching in Reconfigurable Logic

Kubilay Atasu
IBM Research - Zurich

Presented at FPT 2015
Queenstown, NZ