Accelerating Text Analytics Queries on Reconfigurable Platforms

Kubilay Atasu, Raphael Polig, Christoph Hagleitner, H. Peter Hofstee
Laura Chiticariu, Frederick Reiss, Huaiyu Zhu, Cesar Berrospi

June 14, 2015 @ CARL Workshop
Outline

- Introduction
- Text analytics use cases
- SystemT text analytics software
- HW-accelerated SystemT
- HW-accelerated regex matching
- Conclusions
The importance of performance

Financial Data
- Regulatory filings can be in tens of millions and several TBs

Machine data
- 1GB of app server logs per day
- A medium-size data center has tens of thousands of servers → Tens of Terabytes of system logs per day

Social Media Data
- 1+ TB Twitter data per day
- 400+ TB per year

Source: UCSC Lecture on Information Extraction by F. Reiss, L. Chiticariu, Y. Li, 2014
Big Data image by Camelia Boban; Social Media image by Yoel Ben-Avraham

© 2015 IBM Corporation
Spectrum of throughput

- The more complex the task, the slower the runtime performance
- But the higher the information accuracy

Source: UCSC Lecture on Information Extraction by F. Reiss, L. Chiticariu, Y. Li, 2014
Coherent Accelerator Processor Interface (CAPI)

Virtual Addressing
- Accelerator can work with same memory addresses that the processors use
- Pointers de-referenced same as the host application
- Removes OS & device driver overhead

Hardware Managed Cache Coherence
- Enables the accelerator to participate in “Locks” as a normal thread
- Lowers Latency over IO communication model

Customizable Hardware Application Accelerator
- Specific system SW, middleware, or user application
- Written to durable interface provided by PSL

Processor Service Layer (PSL)
- Present robust, durable interfaces to applications
- Offload complexity / content from CAPP

Virtual Addressing
- Accelerator can work with same memory addresses that the processors use
- Pointers de-referenced same as the host application
- Removes OS & device driver overhead

Hardware Managed Cache Coherence
- Enables the accelerator to participate in “Locks” as a normal thread
- Lowers Latency over IO communication model

Customizable Hardware Application Accelerator
- Specific system SW, middleware, or user application
- Written to durable interface provided by PSL

Processor Service Layer (PSL)
- Present robust, durable interfaces to applications
- Offload complexity / content from CAPP
Hardware-accelerated text analytics

IBM InfoSphere BigInsights

AQL Query

SystemT Text Analytics Compiler & Runtime

POWER8

CAPI

FPGA

IBM InfoSphere Streams

© 2015 IBM Corporation
Outline

- Introduction
- **Text analytics use cases**
  - SystemT text analytics software
  - HW-accelerated SystemT
  - HW-accelerated regex matching
- Conclusions
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 is a super-important shift for us in terms of code access.”

Richard Stallman, founder of the Free Software Foundation, countered saying …
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 is a super-important shift for us in terms of code access.”

Richard Stallman, founder of the Free Software Foundation, countered saying …
Text analytics in IBM Crystal+: news search services

- News Search Engine
  - Relevant docs
  - News search query

- Web Application Server
  - Web search query
  - Annotated docs

- Intel Blade
  - Document search query
  - Annotated docs

- Watson Content Analytics (Document Processor)

- Crawlers

- User Interface

- Web Sources
Acceleration of Crystal+ news search services

News Search Engine

Web Application Server

Document Processor

User Interface

JAVA

CAPI

FPGA

POWER 8

Produce results in real time!
Outline

- Introduction
- Text analytics use cases
- SystemT text analytics software
- HW-accelerated SystemT
- HW-accelerated regex matching
- Conclusions
SystemT overview

AQL

rule language with familiar SQL-like syntax
specify annotator semantics declaratively

systemT optimizer

choose an efficient execution plan that implements the semantics

compiled operator graph

systemT runtime

highly scalable, embeddable Java runtime

input document stream

annotated document stream
The CARL workshop is really awesome!

Character start offset | Character end offset | Token start id | Token end id

Conference name <Span> | Rating <Span> | Count <Int>

Schema
A simple SystemT information extraction rule

- Find the names (regex) that are at most 20 chars after a title (dict.)
... Tomorrow, we will meet Mark Scott, Howard Smith, ...

**Annotation Operator Graph (AOG)**

1. Regex `<Caps>`
2. Dictionary `<First>`
3. Dictionary `<Last>`
4. Join `<First> <Caps>`
5. Join `<First> <Last>`
6. Union
7. Consolidate

**Keywords:**
- Tomorrow
- Mark
- Scott
- Howard
- Smith
- Mark
- Scott
- Howard
- Smith
- Mark
- Scott
- Howard
- Smith
- Mark
- Scott
- Howard
- Smith
- Mark
- Scott
- Howard
- Smith
- Mark
- Scott
- Howard
- Smith
- Mark
- Scott
- Howard
- Smith
- Mark
- Scott
- Howard
- Smith
- Mark
- Scott
- Howard
- Smith
AOG of a real-life SystemT IE query
Outline

- Introduction
- Text analytics use cases
- SystemT text analytics software
  - HW-accelerated SystemT
- HW-accelerated regex matching
- Conclusions
SystemT acceleration: Partitioned system

POWER processor

- Regex
- Difference
- Union
- Consolidate

System memory

- Document buffer
- Result buffer A
- Result buffer B

FPGA

- LOAD
- Job Control
- STORE

Input unit

Dictionary

Output unit

Regex

Join
Multithreaded communication scheme

SystemT main thread

SystemT worker thread

Wait for results

SystemT worker thread

SUBMIT (via MMIO write)

HW scheduler dispatches jobs to any free stream

FPGA stream

Wait for results

POLL (status in memory)

Results and status are written to memory

J A V A

J N I

F P G A

© 2015 IBM Corporation
Elastic hardware interface

Diagram showing the interaction between a producer and a consumer through a schema. The schema includes spans 1, 2, and 3, with start and end offsets represented as 32b and start tokens as 16b.
- Measurements on a two socket POWER7 server with 8 cores per CPU @3.55GHz

The evaluated queries are completely offloaded to FPGA logic (no partitioning)
The evaluated queries are completely offloaded to FPGA logic (no partitioning)
FPGA resource utilization: Effect of reducing the offset width

- Basic data structures: spans and schemas

<table>
<thead>
<tr>
<th>Character start offset</th>
<th>Character end offset</th>
<th>Token start id</th>
<th>Token end id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company name</td>
<td>Suppliers</td>
<td># Investors</td>
<td></td>
</tr>
</tbody>
</table>

- Scalability is limited by BRAM & register usage
- 4 pipelines: 4 bytes/cycle
- 200 MHz → 800 MB/s
Live demo of Crystal+ acceleration on POWER8

- Higher SW performance
- CAPI system
- Virtual addressing from user FPGA
- Multiple cards per CPU

Stratix V GX A7

POWER 8
Outline

- Introduction
- Text analytics use cases
- SystemT text analytics software
- Hardware-accelerated SystemT
- **HW-accelerated regex matching**
- Conclusions
Consider the regex .*(a|b|aa|aba)

Can be transformed into NFA/DFA

Traditional architectures do not support start offset reporting & leftmost matching:
- Reconfigurable NFAs (Sidhu FCCM 2001, Bispo FPT 2006, Yang ANCS 2008)
- Programmable DFAs (Smith SIGCOMM 2008, Van Lunteren MICRO 2012)
Finding leftmost regular expression matches

- Assume that we are searching for the regex \(.*(a|aa|aaaa)\) in the input string “aaaa”
- Find the regex match with the smallest start offset value at each end offset position
- The leftmost matches are marked using solid lines

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>'a'</td>
<td>'a'</td>
<td>'a'</td>
<td>'a'</td>
</tr>
</tbody>
</table>
```

- matches found for 'a'

```
(0,0) (1,1) (2,2) (3,3)
```

- matches found for 'aa'

```
(0,1) (1,2) (2,3)
```

- matches found for 'aaaaa'

```
(0,3)
```
Contributions of this work

1. Extending Sidhu and Prasanna’s NFA architecture to support start offset reporting
2. A graph coloring based register clustering method to minimize the register usage
3. An efficient leftmost match computation method without using offset comparisons

Kubilay Atasu: Resource-efficient regular expression matching architecture for text analytics. ASAP 2014.
Extending Sidhu & Prasanna’s architecture

- Add a start offset register to each NFA state
- `offset_reg[0]` = value of current offset position
- DRAWBACK: redundant start offset registers
Clustering offset registers

- Build a conflict graph and apply graph coloring
- States with the same color can share registers
Leftmost match computation

- Assume that state 0 and state 1 are active and the current input is “a”
- We have to compute $\text{offset\_reg}[2] = \text{MIN}(\text{offset\_reg}[0], \text{offset\_reg}[1])$
Experiments (L7 filter regexs)

- Altera Stratix IV GX530KH40C2, Altera Quartus II V11 tools
- 32-bit start offset registers, 250 MHz target clock frequency
- NFA representation: Follow Automata with character classes
- Scalability: 1000 regexs with start offset reporting on FPGAs

Kubilay Atasu: Resource-efficient regular expression matching architecture for text analytics. ASAP 2014.
Outline

- Introduction
- Text analytics use cases
- SystemT text analytics software
- Hardware-accelerated SystemT
- HW-accelerated regex matching
- Conclusions
Conclusions

- A prototype system that accelerates execution of text analytics queries by
  - utilizing POWER processors and an Altera Stratix FPGAs
  - defining a flexible HW/SW interface with multi-threading support
  - automating generation of query-specific hardware accelerators

- Up to 79x higher document processing throughput vs multi-threaded SW
  - up to 85x better system energy efficiency vs. POWER 7 processor

- Scalable regular expression accelerator that supports advanced features

- Live demonstration of Crystal+ news search acceleration on POWER8

- Ongoing work: programmable overlay architectures to avoid re-synthesis
  - enables support for interactive and complex user queries
QUESTIONS?

Related Publications

- **Architecture & Hardware Compiler:**

- **Regular Expression Matching:**

- **Dictionary Matching:**