Accelerating search in data centers and the cloud

Gustavo Alonso
Systems Group
Department of Computer Science
ETH Zurich, Switzerland
High Performance Software and Systems

• **Algorithms**: What can be done if we are not (or less) bound by the limitations of modern CPUs?

• **Architectures**: What can be done if we are not (or less) bound by the limitations of current Von Neumann and x86 style architectures?

• **Systems**: If we are no longer bound by CPU and architectural limitations, how would complete systems look like?
This talk ...

... is about how to use hardware accelerators (FPGAs) in a meaningful way in the context of search engines.

Emphasis is not only on the hardware solution but on the need to approach hardware acceleration as a hardware-software co-design issue.

Algorithms -> Architecture -> Systems
Overview

• Use Case: Flight Search
• Latency vs Throughput in modern data centers
• Route scoring
  • Problem statement
  • Decision trees ensemble
  • Hardware acceleration
• Minimum connection time
  • Problem statement
  • Rule engines
  • Hardware Acceleration
• Discussion and ideas
• Equipment used in this work has been donated by Intel (HARP), Microsoft (Catapult), and Xilinx (UltraScale cards)
• Work done in collaboration with Amadeus
Searching for flights
Search Engines Today

- Many of us have a naïve notion of how search engines work
- Nowadays, search engines are complex distributed systems:
  - Large scales (hundreds, thousands of computing nodes)
  - Highly specialized components
  - Redundancy for fault tolerance
  - Highly tuned performance of every stage
  - Costly to operate and maintain
- Scalability is solved today through brute force: throw more computing nodes to the problem => neither efficient nor effective
Amadeus current architecture: Master Pricer flow

- Several hundred, multicore servers involved in each of the stages of the search process
- Overall latency limit is 4 seconds (interactive search)
- Throughput determines the capacity of the systems AND limits the quality of the results
Latency vs Throughput
Basics of distributed system design

• SLA is defined as a latency bound with a minimum throughput
• How to improve throughput?
  • Scaling out and parallelizing

• How do you improve latency?
  • Making computing nodes faster => no longer works
  • Avoiding overheads => the opposite of what we do to improve throughput
  • Do less => compromise on quality of the results to finish on time
Route Scoring
Problem statement

• **Improve the quality** of the results
  • Quality can be improved by considering more routes and selecting them better in the Route Scoring module ...
  • ... but must be done **without increasing latency**

• **Increase efficiency** of the overall system
  • Reduce network hops
  • Reduce number of servers involved
  • Consolidate operations in one node instead of several
  • ... but must be done **without decreasing throughput**

• **Budget for Route Scoring is 10 ms**
Decision trees

M. Owaida et al. FPL’17, FPL’18
Application Partitioning on FPGA Clusters: Inference over Decision Tree Ensembles
Route Scoring on an FPGA


## Results Route Scoring on the cloud

<table>
<thead>
<tr>
<th>AWS Instance</th>
<th>Features</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU P2.xlarge</td>
<td>1 NVidia K80</td>
<td>0.90 $/hour</td>
</tr>
<tr>
<td>GPU P3 2xlarge</td>
<td>1 NVidia V100</td>
<td>3.06 $/hour</td>
</tr>
<tr>
<td>CPU C5 2xlarge</td>
<td>8 vCPUs</td>
<td>0.34 $/hour</td>
</tr>
<tr>
<td>FPGA F1 2xlarge</td>
<td>1 Virtex UltraScale+</td>
<td>1.65 $/hour</td>
</tr>
</tbody>
</table>

### On-premise

<table>
<thead>
<tr>
<th>Instance</th>
<th>Features</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP ProLiant</td>
<td>56 CPU cores</td>
<td>11K  $</td>
</tr>
<tr>
<td>Intel’s HARP v2</td>
<td>1 Arria 10 FPGA</td>
<td>7.5K $</td>
</tr>
<tr>
<td>Xilinx VCU1525</td>
<td>1 Virtex UltraScale+</td>
<td>7.5K $</td>
</tr>
</tbody>
</table>

![Bar chart showing comparison between AWS and on-premise instances](chart.png)
Amadeus possible architecture: Route Scoring on FPGA

The FPGA based Route Scoring module is used to:
- Score more routes (improving quality of the results)
- Merge the scoring with another stage (domain explorer)
- Break the latency vs throughput tradeoff
- Reduce the number of CPUs and nodes needed
Minimum Connection Time
The Minimum Connection Time (MCT) is used in the Domain Explorer to produce valid routes:

- Determines how many results can be produced for routes with one or more hops
- Advantages of new Route Scoring bound by the ability to compute the MCT
- MCT takes 37% of the overall CPU time of the Domain Explorer
- One of the reasons why several hundred nodes are needed to parallelize the Domain Explorer
Problem statement

• MCT currently implemented using Business Rule Engine
• Rules applied are updated regularly
• Very expensive to scale in terms of queries/second
• CPU and memory intensive workload
• Needs to support many more rules (currently 136k rules, soon 400k)

• Increase the throughput of the MCT module
Simple example of rules for MCT

<table>
<thead>
<tr>
<th>Rule</th>
<th>Airport</th>
<th>Terminal</th>
<th>Inbound</th>
<th>Outbound</th>
<th>Min. trsf. time</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Zurich ZRH</td>
<td>*</td>
<td>International</td>
<td>*</td>
<td>90 min</td>
<td>Low</td>
</tr>
<tr>
<td>1</td>
<td>Zurich ZRH</td>
<td>*</td>
<td>Schengen</td>
<td>Schengen</td>
<td>25 min</td>
<td>Middle</td>
</tr>
<tr>
<td>2</td>
<td>Zurich ZRH</td>
<td>T3</td>
<td>Schengen</td>
<td>Schengen</td>
<td>40 min</td>
<td>High</td>
</tr>
<tr>
<td>3</td>
<td>Zurich ZRH</td>
<td>T1</td>
<td>Schengen</td>
<td>Schengen</td>
<td>25 min</td>
<td>High</td>
</tr>
<tr>
<td>4</td>
<td>Paris CDG</td>
<td>T1</td>
<td>Schengen</td>
<td>Schengen</td>
<td>25 min</td>
<td>High</td>
</tr>
<tr>
<td>5</td>
<td>Paris CDG</td>
<td>T2F</td>
<td>International</td>
<td>International</td>
<td>45 min</td>
<td>High</td>
</tr>
</tbody>
</table>

A business rule engine runs queries over the set of rules to find out which rules apply to the concrete query. The classical approach is to use the RETE algorithm as implemented in essentially all business rule engines. Amadeus uses Drools and their own implementation of a rule engine (the latter is used for MCT)
NFA-inference Algorithm

Dual Strategy

Observations:
- One pointer per level
- Goes wide and deep as early as possible
- Respects the continuous flow of matching rules
Performance

(b) Throughput in queries per second
Preliminary results

- Resource utilization

<table>
<thead>
<tr>
<th>VCU1525</th>
<th>LUT</th>
<th>REG</th>
<th>BRAM</th>
<th>URAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CU</td>
<td>6,582 (0.64%)</td>
<td>8,892 (0.41%)</td>
<td>15 (0.81%)</td>
<td>76 (7.92%)</td>
</tr>
<tr>
<td>2 CU</td>
<td>10,764 (1.04%)</td>
<td>13,533 (0.63%)</td>
<td>15 (0.81%)</td>
<td>76 (7.92%)</td>
</tr>
<tr>
<td>Total</td>
<td>1,033,612 (100%)</td>
<td>2,151,665 (100%)</td>
<td>1,859 (100%)</td>
<td>960 (100%)</td>
</tr>
</tbody>
</table>

- Performance
  - Baseline current system: 4 ms per query
  - With our algorithm: ~80-100 us per query (both CPU and FPGA)
  - FPGA in streaming mode will lower the latency even further
  - FPGA better than CPU for large batches of queries (>100)
Discussion and Ideas
Lessons learned = chips not enough

• The key is hardware-software co-design
  • Significant gains in performance and efficiency

• Our research program:
  • Algorithms: new algorithms for non-Von Neumann architectures
  • Architectures: where to put the accelerators and how to connect them
  • Systems: how acceleration fits in the entire stack
A platform for exploration: Enzian

Joint work with Timothy Roscoe (ETHZ)