Deepgreen MPP with FPGA:
A supercharged Greenplum Data Warehouse solution

Presented By

Feng Tian
Founder
It’s Time for a complete rewrite

The End of an Architectural Era
(It’s time for a complete rewrite)

by

Michael Stonebraker

New Application

Rich Data
  - Text
  - IoT, Geospatial
  - Media

Intelligent Data
  - Query getting more complex
  - Geospatial
  - Machine learning/Data mining
  - AI/Deep learning
Complete Rewrite

Hardware Trend

CPU has peaked, FPGA has more room

Storage Hierarchy

- Big memory
- SSD
- Lots of bandwidth

Network
10, 100 GigE
Deepgreen MPP Database

- MPP (Massively Parallel Processing) shared nothing data warehouse
- Based on the open source Greenplum Database, 100% compatible
- Complete new query execution engine (LLVM JIT, SIMD)
- On premise and in clouse (AWS)
- Adding FPGA

More than 3X faster on average!
A New Golden Age for Computer Architecture

- Domain Specific Hardware/Software Co-Design
- Enhanced Security
- Open Instruction Set
- Agile Chip Development

AWARDS & RECOGNITION
John Hennessy and David Patterson Receive 2017 ACM A.M. Turing Award
## Putting FPGA In Deepgreen

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Our Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ Memory is big, but not big enough</td>
<td>➢ Identify the bottleneck</td>
</tr>
<tr>
<td>➢ Throughput vs Latency</td>
<td>➢ New algorithm tuned for FPGA</td>
</tr>
<tr>
<td>➢ Multi-CPU/Core</td>
<td>➢ Offload to FPGA, none preemptive</td>
</tr>
<tr>
<td>➢ Multiuser environment</td>
<td>➢ XLIW: eXtra Long Instruction Word</td>
</tr>
</tbody>
</table>
XLIW: eXtra Long Instruction Word

Kernel

Result

Result

Result

XLIW: Hasher
Data, Data, Data …

XLIW: Hasher
Data, Data, Data …
# Use Case 1: Hash Join

**Hash Join**

- Select * FROM A JOIN B ON A.x = B.x and A.y = B.y …
- One of the most important, expensive operation in OLAP
- Very simple algorithm
  - Read everything from A (or B, whichever is smaller)
  - Build a hash table
  - For each record from B
  - Probe the hash table.
  - Out all matching pairs
  - More complicated in real system, but this is the idea
- Lots of records joined
- Hash table is not cache friendly

**XLIW for Hash Join**

- Pack a lot of records of A, send to FPGA to compute hashes
- Instead of using hash table, we sort the hashes using a very fast radix sort. (10x faster than quicksort)
- Pack a lot of records of B, send to FPGA to compute hashes
- Sort hashes of B
- Merge
- It is a hybrid hash/sort merge join
Case 1: Hash Join Performance

TPCH 100G on AWS F1

Q17
- Greenplum: 250
- Deepgreen: 80
- Deepgreen + XLIW: 0

Q26
- Greenplum: 150
- Deepgreen: 30
- Deepgreen + XLIW: 0
Use Case 2: GeoSpatial Join
Use Case 2: GeoSpatial Join

- SELECT area, count(*) FROM point JOIN area
  
  WHERE ST_Intersects(point, area)

  group by area

- How many user/devices (points) in each area (polygon)

- Intersects is an expensive operation and forces a nested loop join (slow)
  
  > Naïve approach will never finish
Use Case 2: GeoSpatial Join

Greenplum + PostGIS

- Build index (R-tree)
- Index Nestloop Join
  - For each polygon, using index to lookup points nearby
  - Check the intersects condition
- Could take hours

GeoSpatial Join + XLIW

- Do not use index
- Scan outer loop, build an in-memory data structure
  - Still expensive operation, but cheaper than compute intersection (like building an R-tree)
- Scan inner loop, probing the in memory data structure (like probing R-tree)
- Check intersection
  - This step is dominating execution time
XLIW: GeoSpatial Operations

> For Intersects

> Packing many (point, area) pairs, send to FPGA, compute result
  >> We are not so worried about serialization cost this time

> We could have let FPGA build the in memory data structure for us
  >> Currently not the bottleneck
Use Case II: Performance

Spatial Join

- Greenplum
- SpatialJoin
- SpatialJoin-XLIW

1Mx1M vs 10Mx10M
Use Case 3: Adding Intelligence

- An XLIW for data mining/machine learning

- Deepgreen Transducer Framework
  - Allow user to embed C/Java/Go/Python code in SQL
  - Interleaved with SQL Engine code
  - First class citizen, optimized by query optimizer, executed in parallel, streaming data to/from SQL query operators like Sort/Join/Aggregate

- ML libraries, Tensor Flow
  - For example, Deep Neural Network in FPGA
Current Status and Future Directions

▶ Deepgreen DB Appliance on AWS F1
  ▶ See our demo
  ▶ On AWS Market Place soon

▶ On premise

▶ We are just scratching the surface
  ▶ More use cases, endless opportunities
  ▶ More to squeeze
Conclusion and Thank you

- Deepgreen MPP with FPGA on AWS F1 or On Premise
- Built for petabyte-size data warehouse applications
- Taking full advantage of modern hardware and FPGA, many crucial queries can be executed swiftly, increasing productivity of data scientists.

- Thank Xilinx team!
- Thank you all!
Adaptable.
Intelligent.

VITESSE DATA