



Qizmt Open Source Paradigms

<http://qizmt.myspace.com/>

Version: Alpha 1.2

Revision: 2010-02-02



Qizmt Mapreduce

- Robust C# mapreduce engine, fast recovery on disk failure
- Largest cluster stress-tested is 1808 cores sorting 1TB of 100-byte rows in 5 minutes, 10TB of 100-byte rows in 2hrs

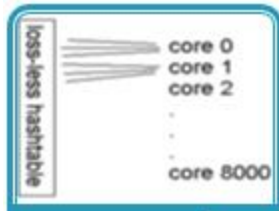
SQL to Mapreduce Translation – Qizmt SQL Extension

- Scalable SQL to Mapreduce translation with over ADO.NET
- Most SQL functionality supported

SQL Distributed Memory – Qizmt SQL Extension

- Mutable distributed memory with SQL-like access over ADO.NET (limited INSERT, DELETE and SELECT only)
- Optimal for both off-line graph processing and near-real-time graph traversals
- INSERT maintains the current replication level of the cluster
- Because it is built on top of mapreduce framework, partitioning is trivial and scales well past Neo4J-oriented solutions
- Largest cluster stress-tested is 4-billion node graph on 512 cores with 2TB of distributed memory and 64Gb/s memory-bandwidth, completed two 2 levels of traversal from each node in 7 hours.

Qizmt Mapreduce



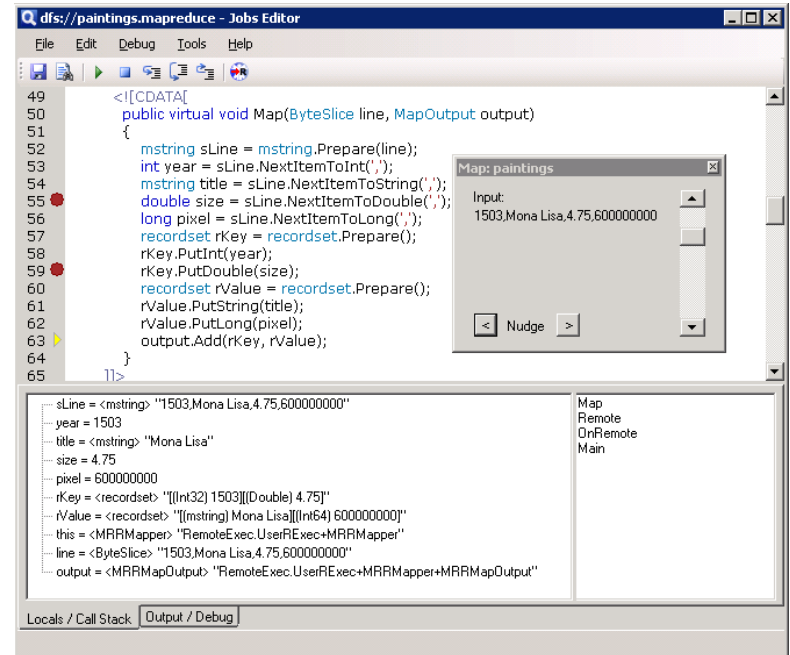
Competitive Speed



Redundancy / Failover / Scalability



Built-in IDE/Debugger



```

49 <![CDATA[
50     public virtual void Map(ByteSlice line, MapOutput output)
51     {
52         mstring sLine = mstring.Prepare(line);
53         int year = sLine.NextItemToInt(',');
54         mstring title = sLine.NextItemToString(',');
55         double size = sLine.NextItemToDouble(',');
56         long pixel = sLine.NextItemToLong(',');
57         recordset rKey = recordset.Prepare();
58         rKey.PutInt(year);
59         rKey.PutDouble(size);
60         recordset rValue = recordset.Prepare();
61         rValue.PutString(title);
62         rValue.PutLong(pixel);
63         output.Add(rKey, rValue);
64     }
65 ]>

```

Map: paintings

Input:
1503.Mona Lisa,4.75,600000000

< Nudge >

```

sLine = <mstring> "1503.Mona Lisa,4.75,600000000"
year = 1503
title = <mstring> "Mona Lisa"
size = 4.75
pixel = 600000000
rKey = <recordset> "[[Int32] 1503][[Double] 4.75]"
rValue = <recordset> "[[mstring] Mona Lisa][[Int64] 600000000]"
this = <MRRMapper> "RemoteExec.UserRExec+MRRMapper"
line = <ByteSlice> "1503.Mona Lisa,4.75,600000000"
output = <MRRMapOutput> "RemoteExec.UserRExec+MRRMapper+MRRMapOutput"

```

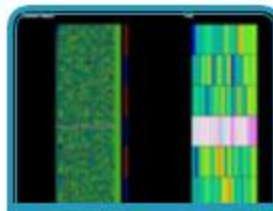
Map Remote OnRemote Main



Delta-Only Exchange Option



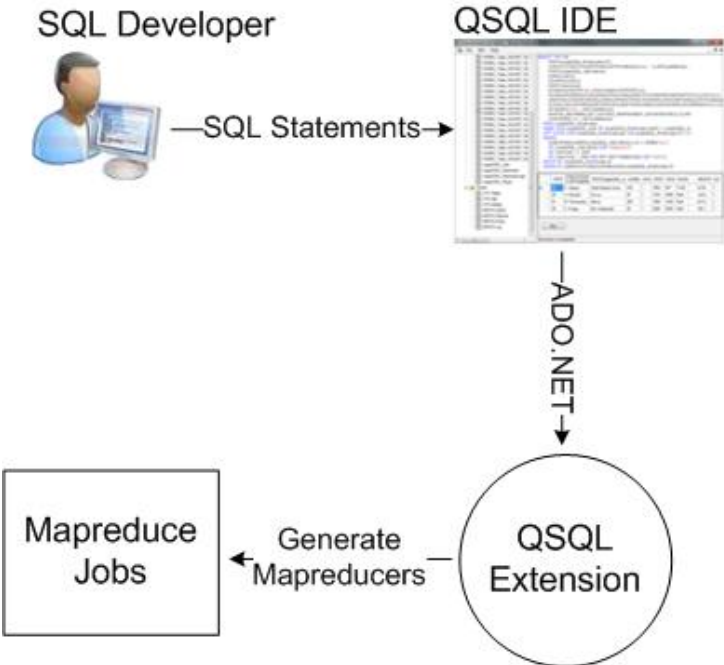
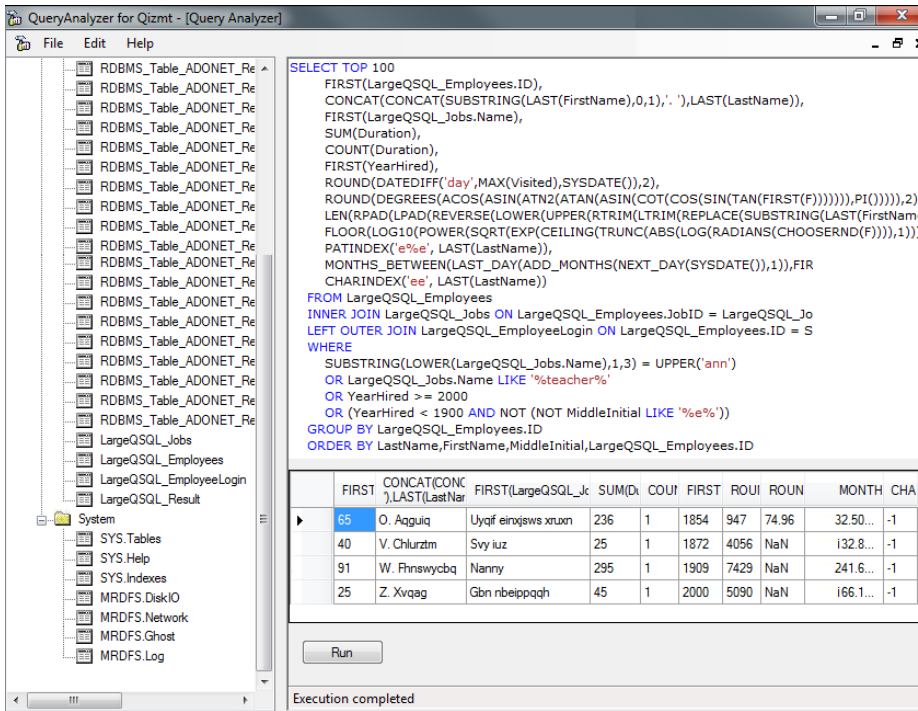
Command From Any Machine



No Sampling Entropy for Sorting

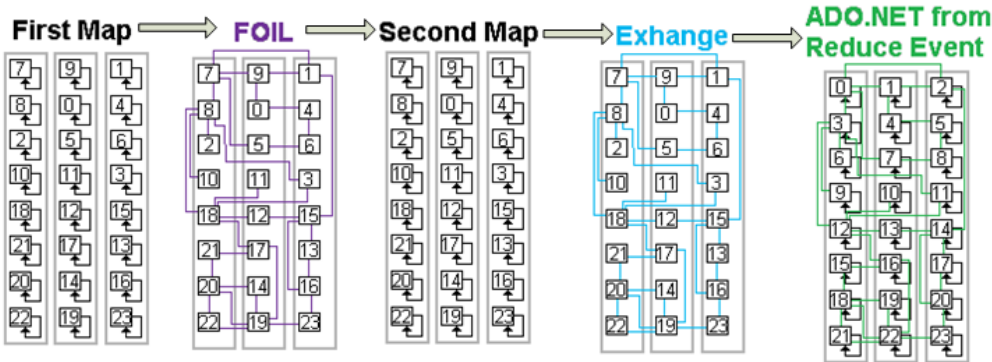
- Has been in use for one year with no data loss on any in-house cluster with replication of 2 or greater
- Average cluster down-time on disk failure is about 1 hour and includes notification of disk failure, removal of failed machine, adding replacement machine to cluster and redistributing the data.
- Active participation in the open source community since the GPLv3 releases were started in September 2009
 - <http://code.google.com/p/qizmt/updates/list>

SQL to Mapreduce Translation



- Most SQL statements supported.
- QSQL statements are standard SQL, no custom partitioning or cluster-oriented commands required.
- SQL statements may be executed ad-hoc via QSQL IDE or automated with ADO.NET.
- SQL statements are translated into brute-force but highly scalable mapreducers.
- The replication factor of the underlying Qizmt cluster is inherited.

SQL Distributed Memory Performance Test



```
CREATE RINDEX indPaintings
FROM tblPaintings ON artistID
```

```
RSELECT * FROM indTest WHERE KEY = 1 OR KEY = 3 ...
```

```
BEGIN BULKINSERT
```

```
RDELETE FROM indPaintings WHERE KEY = 1
```

```
RINSERT INTO indPaintings VALUES (2, 'D') WHERE KEY = 2
```

```
RDELETE FROM indPaintings WHERE KEY = 3 AND name = 'Da Vinci'
```

```
END BULKUPDATE
```

RSELECT stress test results on 512 cores with 64Gb/s memory bandwidth

Maximum Rels per node	Average Rels per node	Nodes In Graph	Runtime	OR Statements per second	Graph Traversals per second
5	2.5	2,000,000,000	2 hours	751,314	2,930,127
5	2.5	4,000,000,000	7 hours	396,825	1,547,619
6,400	3,200	10,000	17 minutes	31,158	99,739,045
300	150	200,000,000	47 hours	176,825	26,701,795

- Each node consisted of an integer with a set of related integers.
- Each node is visited, and a full traversal is made to all related nodes as well as all relationships of relationships.
- In addition to the 2-level traversal, these tests also include the time it takes to evenly partition, index and load the graph into distributed memory from a randomly ordered set of relationships in a cluster with redundancy 2.
- Although these stress tests used random data, we have verified that similar performance holds for highly skewed user data as the physical location of any node in the distributed graph is entirely random.
- Both RSELECT and BULKINSERT failover during execution if a machine is lost during operation.