Trinity: A Distributed Graph Engine on a Memory Cloud

Bin Shao, Yatao Li, Wei-Ying Ma
Microsoft Research Asia
Why do we need a graph system?
Existing Systems

• Mature data processing systems
  – RDBMS
  – Map Reduce Systems, e.g. cosmos

• Systems specialized for certain graph operations:
  – PageRank
Graph Data is “Special” ...

• Random access (Poor Locality)
  – For a node, its adjacent nodes’ content cannot be accessed without “jumping” no matter how you represent a graph
  – Not cache-friendly, data reuse is hard

• Unstructured nature of graph
  – Difficult to extract parallelism by partitioning data
  – Hard to get an efficient “Divide and Conquer” solution
Graph in the Jail of Storage

- RDBMS/cosmos, mature but not for graphs

- The commonest graph operation “traversal” incurs excessive amount of table joins
Challenge I: Diversity of Graphs

Do we need to design algorithms for each type of graphs?
Challenge II: Diversity of Computations

• Online query processing
  – Shortest path query
  – Subgraph matching query
  – SPARQL query
  – ...

• Offline graph analytics
  – PageRank
  – Community detection
  – ...

• Other graph operations
  – Graph generation, visualization, interactive exploration, etc.

Do we need to implement systems for each graph operation?
Challenge III: The **Scale** of Graphs

Makes most graph algorithms in textbooks ineffective!

<table>
<thead>
<tr>
<th># of Nodes</th>
<th># of Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>US Road Map</td>
<td>million</td>
</tr>
<tr>
<td>Linked Data</td>
<td>billion</td>
</tr>
<tr>
<td>Facebook</td>
<td>trillion</td>
</tr>
<tr>
<td>Satori</td>
<td>billion</td>
</tr>
<tr>
<td>Web</td>
<td>trillion</td>
</tr>
</tbody>
</table>
Trinity Research Roadmap

- **Applications**
  - Real-time knowledge serving on knowledge graph, academic search, etc
- **Algorithms**
  - Subgraph matching, Trinity.RDF, distance oracle, graph partitioning, reachability …
  - [VLDB 2012, 2013, 2014], ICDE 2014
- **Programming models**
- **Online Query Processing**
- **Offline Graph Analytics**
- **Storage infrastructure**
- **Trinity Graph Engine:**
  - [Sigmod 2012, 2013]
- **Trinity Memory Cloud**
Design Philosophy

Not a one-size-fits-all graph system, but a graph engine

Flexible data and computation modeling capability

Trinity can morph into
a large variety of graph processing systems

Trinity = Graph Modeling Tools +
Distributed In-memory Data Store +
Declarative Programming Model
Design Rationale of Memory Cloud

Random access challenge

RAM capacity limit of single machine

Fast random access

Parallel computation

Low latency online query processing

High throughput offline analytics
## System Stack

<table>
<thead>
<tr>
<th>Graph APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetInlinks(), Outlinks.Foreach(...) etc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Graph Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trinity Specification Language</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Cloud (Distributed Key-Value Store)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed Memory Storage</td>
</tr>
<tr>
<td>Message Passing Framework</td>
</tr>
</tbody>
</table>
One Byte Counts
(Trinity vs. PBGL)

Execution Time

Memory Usage
Trinity Specification Language

Graph Modeling
OMG IDL
ICE Slice
TSL
Google ProtoBuf
Message Passing Modeling
Data interchange Format Specification
Why TSL?

• TSL allows users to define graph schemata, and communication protocols through declarative interfaces.

• TSL makes Trinity memory cloud beyond a key-value store
  – Users are allowed to freely define the data schema
  – TSL makes message passing programming ever so easy
Modeling a Movie and Actor Graph

```csharp
[CellType: NodeCell]
cell struct Movie
{
    string Name;
    [EdgeType: SimpleEdge, ReferencedCell: Actor]
    List<long> Actors;
}

[CellType: NodeCell]
cell struct Actor
{
    string Name;
    [EdgeType: SimpleEdge, ReferencedCell: Movie]
    List<long> Movies;
}
```
TSL-enabled Cell Accessor: Efficient and User-friendly

TSL Script

```csharp
using(var cell = UseMyCellAccessor(cellId))
{
    int Id = cell.Id; //Get the value of Id
    cell.Links[1] = 2; //Set Links[1] to 2
}
```

Blob View

```
00000001 00000000 00000000 00000000 00000011 00000000 00000000 00000000
00000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000100 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000011 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```
Modeling Message Passing

```cpp
struct MyMessage {
    string Text;
}

protocol Echo {
    Type: Syn;
    Request: MyMessage;
    Response: MyMessage;
}
```
TSL-Powered Message Passing
Trinity-enabled Graph Computation Paradigms

• **Vertex-centric graph analytics**
  – Prosperous since Pregel, e.g. Giraph, GraphChi

• **Approximate graph computation based on local sampling**
  – Enabled by randomly partitioned in-memory graph
  – Fast approximate computation with minimum communication costs
  – Application: distance oracle [VLDB 2014]

• **Index-free real-time online query processing**
  – Enabled by fast in-memory distributed graph exploration
  – Examples, subgraph match (vldb 2012) and Trinity.RDF (vldb 2013)
### Query Index Examples

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Index Size</th>
<th>Index Time</th>
<th>Update Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ullmann [Ullmann76], VF2 [CordellaFSV04]</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RDF-3X [NeumannW10]</td>
<td>O(m)</td>
<td>O(m)</td>
<td>O(d)</td>
</tr>
<tr>
<td>BitMat [AtreCZH10]</td>
<td>O(m)</td>
<td>O(m)</td>
<td>O(m)</td>
</tr>
<tr>
<td>Subdue [HolderCD94]</td>
<td></td>
<td>Exponential</td>
<td>O(m)</td>
</tr>
<tr>
<td>SpiderMine [ZhuQLYHY11]</td>
<td>-</td>
<td>Exponential</td>
<td>O(m)</td>
</tr>
<tr>
<td>R-Join [ChengYDYW08]</td>
<td>O(nm^{1/2})</td>
<td>O(n^4)</td>
<td>O(n)</td>
</tr>
<tr>
<td>Distance-Join [ZouCO09]</td>
<td>O(nm^{1/2})</td>
<td>O(n^4)</td>
<td>O(n)</td>
</tr>
<tr>
<td>GraphQL [HeS08]</td>
<td>O(m + nd^r)</td>
<td>O(m + nd^r)</td>
<td>O(d^r)</td>
</tr>
<tr>
<td>Zhao [ZhaoH10]</td>
<td>O(nd^r)</td>
<td>O(nd^r)</td>
<td>O(d^L)</td>
</tr>
<tr>
<td>GADDI [ZhangLY09]</td>
<td>O(nd^L)</td>
<td>O(nd^L)</td>
<td>O(d^L)</td>
</tr>
</tbody>
</table>

Index-based subgraph matching [Sun VLDB 2012]
# Query Index Examples

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Index Size for Facebook</th>
<th>Index Time for Facebook</th>
<th>Query Time on Facebook (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ullmann [Ullmann76], VF2 [CordellaFSV04]</td>
<td>-</td>
<td>-</td>
<td>&gt;1000</td>
</tr>
<tr>
<td>RDF-3X [NeumannW10]</td>
<td>1T</td>
<td>&gt;20 days</td>
<td>&gt;48</td>
</tr>
<tr>
<td>BitMat [AtreCZH10]</td>
<td>2.4T</td>
<td>&gt;20 days</td>
<td>&gt;269</td>
</tr>
<tr>
<td>Subdue [HolderCD94]</td>
<td>-</td>
<td>&gt;67 years</td>
<td>-</td>
</tr>
<tr>
<td>SpiderMine [ZhuQLYHY11]</td>
<td>-</td>
<td>&gt;3 years</td>
<td>-</td>
</tr>
<tr>
<td>R-Join [ChengYDW08]</td>
<td>&gt;175T</td>
<td>&gt;10^{15} years</td>
<td>&gt;200</td>
</tr>
<tr>
<td>Distance-Join [ZouCO09]</td>
<td>&gt;175T</td>
<td>&gt;10^{15} years</td>
<td>&gt;4000</td>
</tr>
<tr>
<td>GraphQL [HeS08]</td>
<td>&gt;13T(r=2)</td>
<td>&gt;600 years</td>
<td>&gt;2000</td>
</tr>
<tr>
<td>Zhao [ZhaoH10]</td>
<td>&gt;12T(r=2)</td>
<td>&gt;600 years</td>
<td>&gt;600</td>
</tr>
<tr>
<td>GADDI [ZhangLY09]</td>
<td>&gt;2 \times 10^5T (L=4)</td>
<td>&gt;4 \times 10^5 years</td>
<td>&gt;400</td>
</tr>
</tbody>
</table>

Index-based subgraph matching [Sun VLDB 2012]
Index-free Query Processing
Trinity + RDMS
Data is More and More Connected

• The value of data lies in its *degree of connectedness*
  – “When we search, we’re not just looking for a webpage, we’re looking to **get answers, understand or explore**” – introductory words of Google knowledge graph search

• Relational data management systems are not designed to handle richly connected data

• This is where a native graph system can naturally fit in
From the Perspective of Performance

• For relational databases, *multi-way joins* are needed for handling data of rich connectedness

• In a native graph engine, multi-way join queries can be answered by graph exploration
A Real-life Relational Database Schema
Corresponding Schema Graph
Multi-way Join vs. Graph Traversal

Company
  ID
  ID1 ID2
Problem
  ID
  ID3 ID4
Incident
  ID
  ID

RDBMS
Trinity Graph

Company -> Problem -> Incident
Traversal in A Graph
**SQL**

```
SELECT *
FROM Employee a, EmployeeIncident b, Incident c, IncidentProblem d, Problem e
```

Execution Time: a few seconds

**TSQL**

```
FROM a in {"Employee.FullName='Nikki Dahi' "}
MATCH a(Employee)-->b(Problem)-->c(Incident)
RETURN a, b, c
```

Execution Time: tens of milliseconds
From Perspective of Performance

<table>
<thead>
<tr>
<th>Query</th>
<th>TQL</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>79 ms</td>
<td>834 ms</td>
</tr>
<tr>
<td>Q2</td>
<td>70 ms</td>
<td>778 ms</td>
</tr>
<tr>
<td>Q3</td>
<td>1.3 sec</td>
<td>31min24sec</td>
</tr>
</tbody>
</table>

Real-life Queries on a Microsoft CRM Database
Automated Graph Schema Extraction
Automated Graph Loading
Graph Exploration and Query
Trinity Applications
Source Code Graph (Visual Studio)
Source Code Graph (Visual Studio)
FROM a in {"fullName='Leslie Lamport"} MATCH a-->(Paper:Author:Organization)-->c(Paper) SELECT a.fullName, c.title

Query Result

<table>
<thead>
<tr>
<th>a.FullName</th>
<th>c.Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leslie Lamport</td>
<td>Composition: A Way to Make Proofs Harder</td>
</tr>
<tr>
<td>Leslie Lamport</td>
<td>A Formal Basis for the Specification of Concurrent Systems</td>
</tr>
<tr>
<td>Leslie Lamport</td>
<td>The Operators of TILAC</td>
</tr>
<tr>
<td>Leslie Lamport</td>
<td>The Synchronization of Independent Processes</td>
</tr>
<tr>
<td>Leslie Lamport</td>
<td>Corrigendum: &quot;A New Approach to Proving the Correctness of Multiprocess Programs&quot;</td>
</tr>
<tr>
<td>Leslie Lamport</td>
<td>Comment on Bell's quadratic quotient method for hash coded searching</td>
</tr>
<tr>
<td>Leslie Lamport</td>
<td>SIFT: Design and analysis of a fault-tolerant computer for aircraft control</td>
</tr>
<tr>
<td>Leslie Lamport</td>
<td>Latex: a document preparation system</td>
</tr>
<tr>
<td>Leslie Lamport</td>
<td>Constructing digital signatures from a one-v-ray function</td>
</tr>
<tr>
<td>Leslie Lamport</td>
<td>Specifying</td>
</tr>
</tbody>
</table>
Knowledge Graph
Challenges of Serving Satori

Satori: An ever-growing knowledge repository

<table>
<thead>
<tr>
<th>Raw RDF data</th>
<th>4T+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities</td>
<td>2.4B+</td>
</tr>
<tr>
<td>Triple Facts</td>
<td>20B+</td>
</tr>
</tbody>
</table>

- Complex data schema
  - Rich relations
Challenges of Serving Satori

- Complex data schema
  - Rich relations
  - Multi-typed entities

<table>
<thead>
<tr>
<th>Satori: An ever-growing knowledge repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw RDF data</td>
</tr>
<tr>
<td>Entities</td>
</tr>
<tr>
<td>Triple Facts</td>
</tr>
</tbody>
</table>

```html
123 mso/type.object.name “Pal”
123 mso/type.object.type mso/organism.dog
123 mso/organism.dog.breeds “Collie Rough”
123 mso/type.object.type mso/film.actor
123 mso/film.actor.film 789
789 mso/type.object.type mso/film.film
789 mso/type.object.name “Lassie Come Home”
```

“Pal” as a dog
“Pal” as an actor
Challenges of Serving Satori

Satori: An ever-growing knowledge repository

<table>
<thead>
<tr>
<th>Raw RDF data</th>
<th>4T+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities</td>
<td>2.4B+</td>
</tr>
<tr>
<td>Triple Facts</td>
<td>20B+</td>
</tr>
</tbody>
</table>

- Complex data schema
  - Rich relations
  - Multi-typed entities
- Distributed in-memory knowledge graph

Knowledge Serving Services/APIs

Trinity Graph Engine

Distributed In-memory Graph

Satori in Native Graph Database for Real-time Knowledge Serving
Satori: An ever-growing knowledge repository

<table>
<thead>
<tr>
<th>Raw RDF data</th>
<th>4T+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities</td>
<td>2.4B+</td>
</tr>
<tr>
<td>Triple Facts</td>
<td>20B+</td>
</tr>
</tbody>
</table>

**Challenges of Serving Satori**

- Complex data schema
  - Rich relations
  - Multi-typed entities

Modeling Multi-Typed Satori Entities in a Strongly Typed Manner
Trinity’s Storage Architecture for Satori

Entity Relations
+ In-memory Entity Properties

On-disk Entity Properties

film
actor
director

name

Memory

Disk
Demo
SATORI

Satori Service Portal
Harvard University

Harvard University is an American private Ivy League research university located in Cambridge, Massachusetts, United States, established in 1636 by the Massachusetts legislature. Harvard is the oldest institution of higher learning in the United States and the first corporation (officially The President and Fellows of Harvard College) chartered in the country. Harvard's history, influence...

Types

award.presenting_organization, award.ranked_item, award.winner, book.author, education.academic_institution, education.educational_institution...

Predicates

education.educational_institution.total_enrollment
education.educational_institution.color
education.educational_institution.subsidiary_or_constituent_schools
education.educational_institution.number_of_staff
education.educational_institution.honorary_degrees_awarded
education.educational_institution.school.sports_team

Values

"Harvard Extension School"
"Harvard Medical School"
"Harvard Business School"
"Harvard College"
"Harvard Division of Continuing Education"
"John F. Kennedy School of Government"
Schema Graph

Meta Graph of Satori

Schema Type:

msi/people.person

Go

Schema Path:

msi/people.person

msi/book.author

Fields:

- bust_measurement
- date_of_birth
- eye_color
- first_name
- hair_color
- height
- hips_measurement
- last_name
- waist_measurement
- weight

Links:

- business_employment_tenure
- children
- city_of_birth

Source: <https://www.example.com>
Satori Knowledge Graph Access API

<table>
<thead>
<tr>
<th>API Names</th>
<th>Availability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetEntityIdByName</td>
<td>Available</td>
<td>Gets a list of Trinity entity Ids by the specified entity name.</td>
</tr>
<tr>
<td>GetPredicatesByEntityId</td>
<td>Available</td>
<td>Gets a list of predicates for the entity with the specified Trinity entity Id.</td>
</tr>
<tr>
<td>GetValuesByEntityPredicate</td>
<td>Available</td>
<td>Gets the values of the specified predicates for the specified entity.</td>
</tr>
<tr>
<td>GetSubjectByPredicateObject</td>
<td>Available</td>
<td>Gets the subjects for the given object and a predicate.</td>
</tr>
<tr>
<td>GetEntityIdBySatoriId</td>
<td>Available</td>
<td>Gets the corresponding Trinity entity Id for the specified Satori' Guid.</td>
</tr>
<tr>
<td>GetSatoriIdByEntityId</td>
<td>Available</td>
<td>Gets the corresponding Satori Guid for the specified Trinity entity Id.</td>
</tr>
<tr>
<td>GetRankedEntityIdByName</td>
<td>Available</td>
<td>Gets a list of Trinity entity Ids by the specified entity name sorted by their static rank.</td>
</tr>
<tr>
<td>GetScoredValuesByEntityPredicate</td>
<td>Available</td>
<td>Gets the values of the specified predicates for the specified entity, sorted by confidence score.</td>
</tr>
<tr>
<td>GetSortScoredValuesByEntityPredicate</td>
<td>Available</td>
<td>Gets the values of the specified predicates for the specified entity, sorted by the column index.</td>
</tr>
<tr>
<td>GetEntityDescription</td>
<td>Available</td>
<td>Gets the description of the specified entity Id.</td>
</tr>
</tbody>
</table>

Testing: GetScoredValuesByEntityPredicate

Please input test parameters below:

- EntityId: 24604610039751
- Predicate: mstbnkactor.film

<table>
<thead>
<tr>
<th>PredicateValue</th>
<th>ConfidenceScore</th>
<th>OverallScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>2967469205979</td>
<td>0.73</td>
<td>1.311128</td>
</tr>
<tr>
<td>1.02331907553515</td>
<td>0.71</td>
<td>1.493995</td>
</tr>
<tr>
<td>265920081012109</td>
<td>0.71</td>
<td>1.414611</td>
</tr>
<tr>
<td>501853430450412</td>
<td>0.71</td>
<td>1.710730</td>
</tr>
<tr>
<td>2578565925307352</td>
<td>0.71</td>
<td>1.228659</td>
</tr>
<tr>
<td>576201200425344</td>
<td>0.71</td>
<td>1.272203</td>
</tr>
</tbody>
</table>
Graphical Query Interface

Node Information

Alias: v2
Type: film_director
URI: mso_film_director

Conditions:

Outputs:

You could add some conditions as:

name = Steve S
Tom Cruise

Tom Cruise (born Thomas Cruise Mapother IV; July 3, 1962), is an American film actor and producer. He has been nominated for three Academy Awards and has won three Golden Globe Awards. He started his career at age 19 in the 1981 film Endless Love. After portraying supporting roles in Taps (1981) and The Outsiders (1983), his first leading role was in Risky Business, released in August 1983. Cruise became a full-fledged movie...
Thanks!

http://graphengine.io/