Trinity: A Distributed Graph Engine on a Memory Cloud

Bin Shao

Microsoft Research (Beijing, China)

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

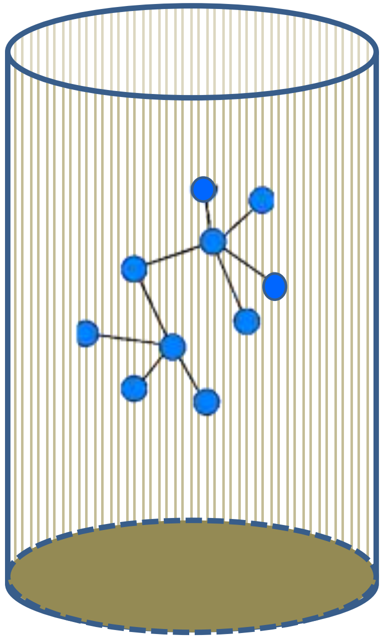
Graph Data is “Special” …

- 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

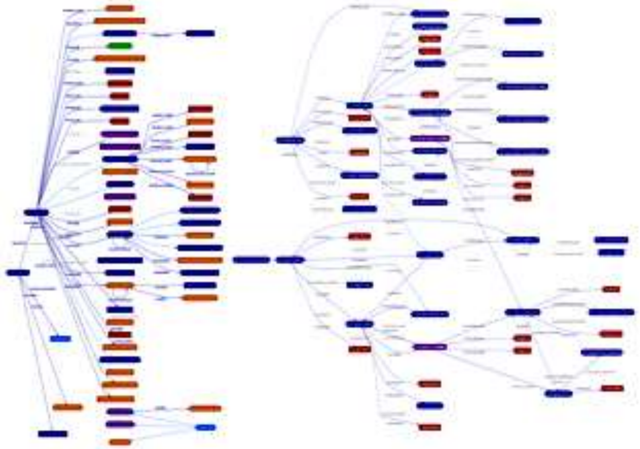
Graph in the Jail of Storage

- RDBMS/cosmos, mature but not for graphs
- The commonest graph operation “traversal” incurs excessive amount of table joins

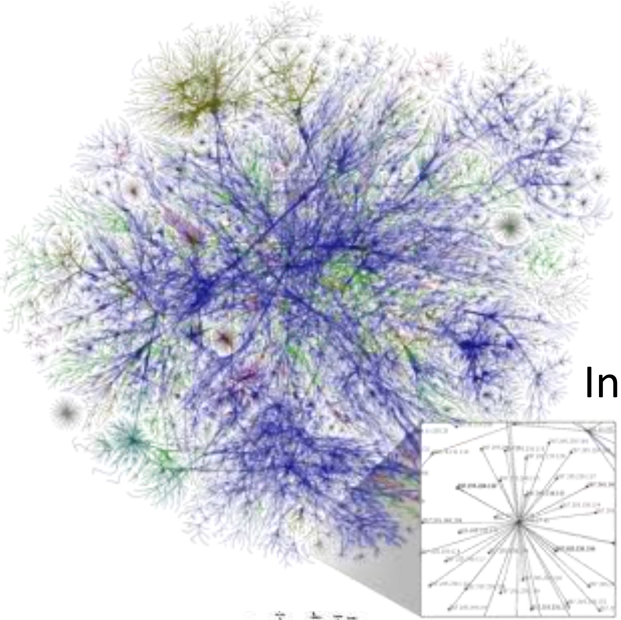


Graph in the
Jail of the storage

Challenge I: Diversity of Graphs

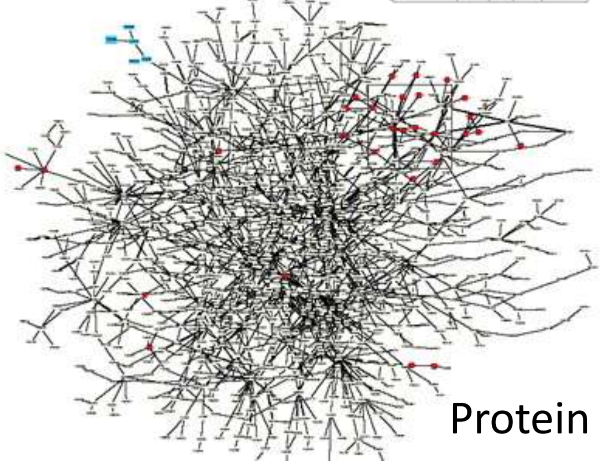


Satori Schema Graph



Internet Web Graph

Social Network



Protein Interaction Network

Do we need to design algorithms for each type of graphs?

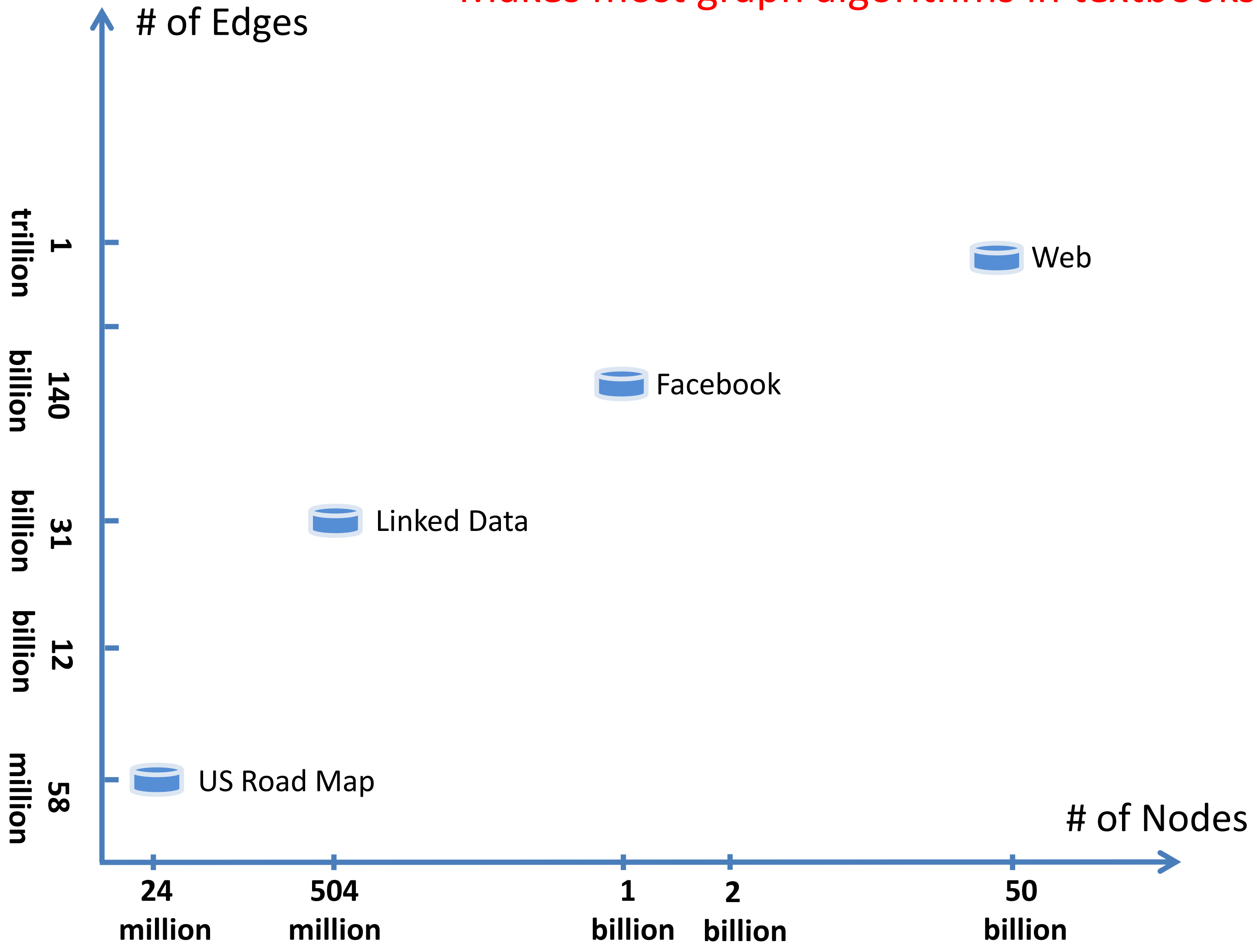
Challenge II: Diversity of Computations

- 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.

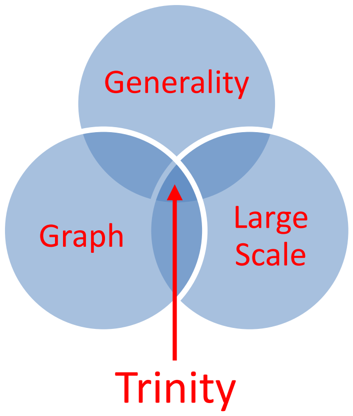
Do we need to implement systems for each graph operation?

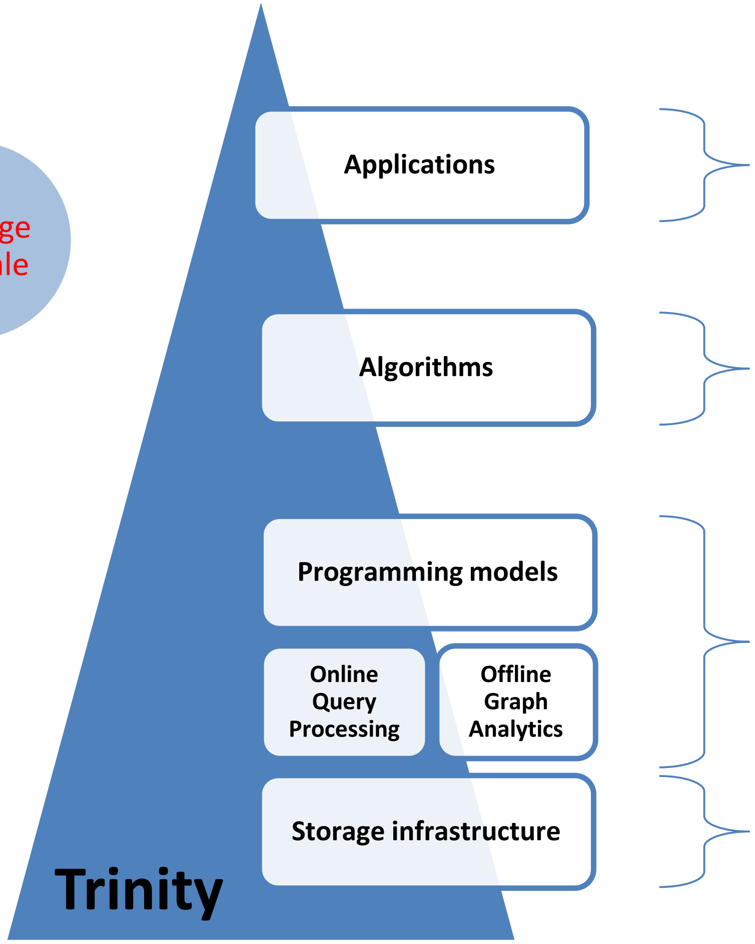
Challenge III: The **Scale** of Graphs

Makes most graph algorithms in textbooks ineffective!



Trinity Research Roadmap





Real-time knowledge serving on knowledge graph, academic search, etc

Subgraph matching, Trinity.RDF, distance oracle, graph partitioning, reachability …
[VLDB 2012, 2013, 2014], ICDE 2014

Trinity Graph Engine:
[Sigmod 2012, 2013]

Trinity Memory Cloud

Design Philosophy

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

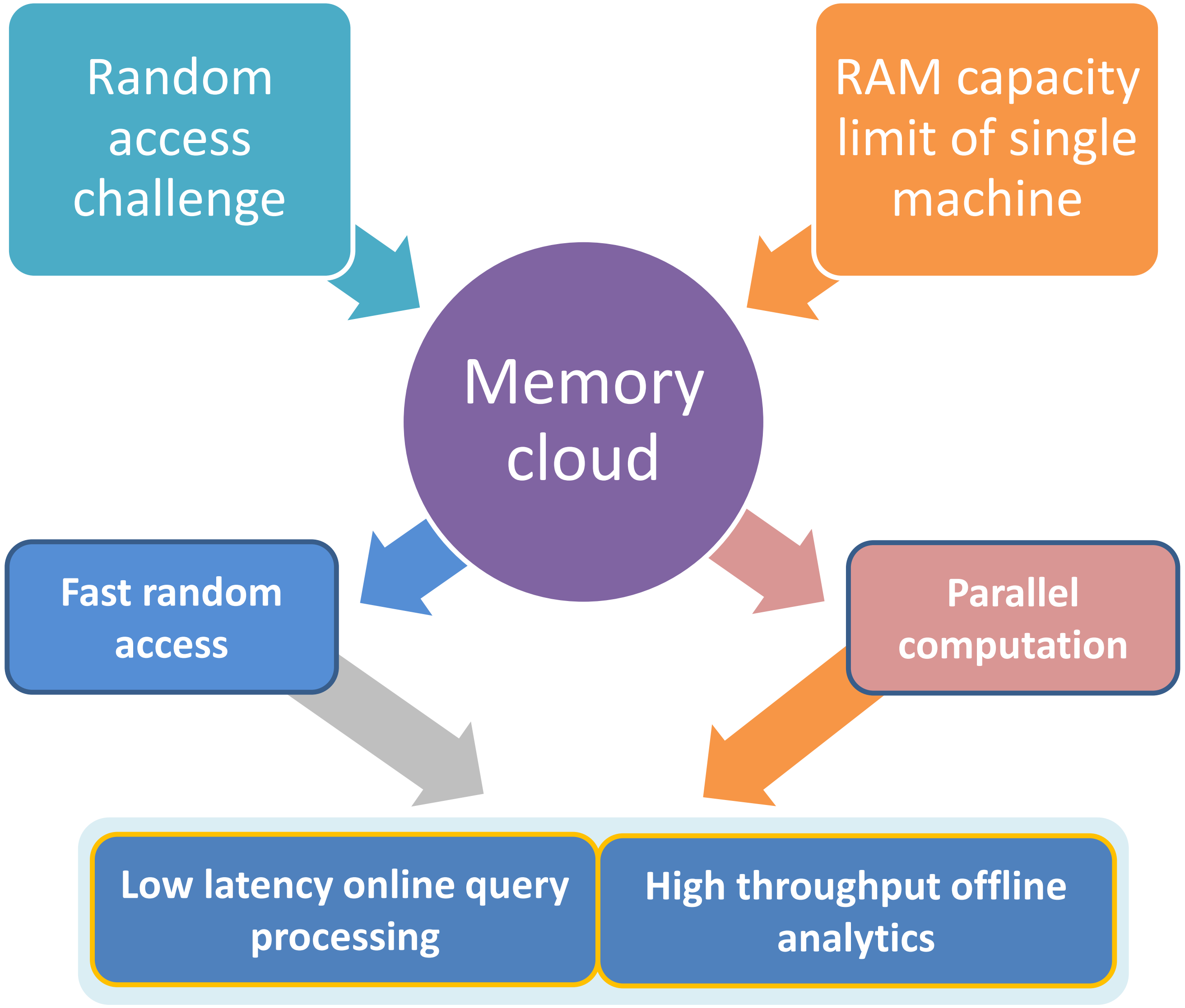
Flexible data and computation modeling capability

⬇

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

Memory cloud

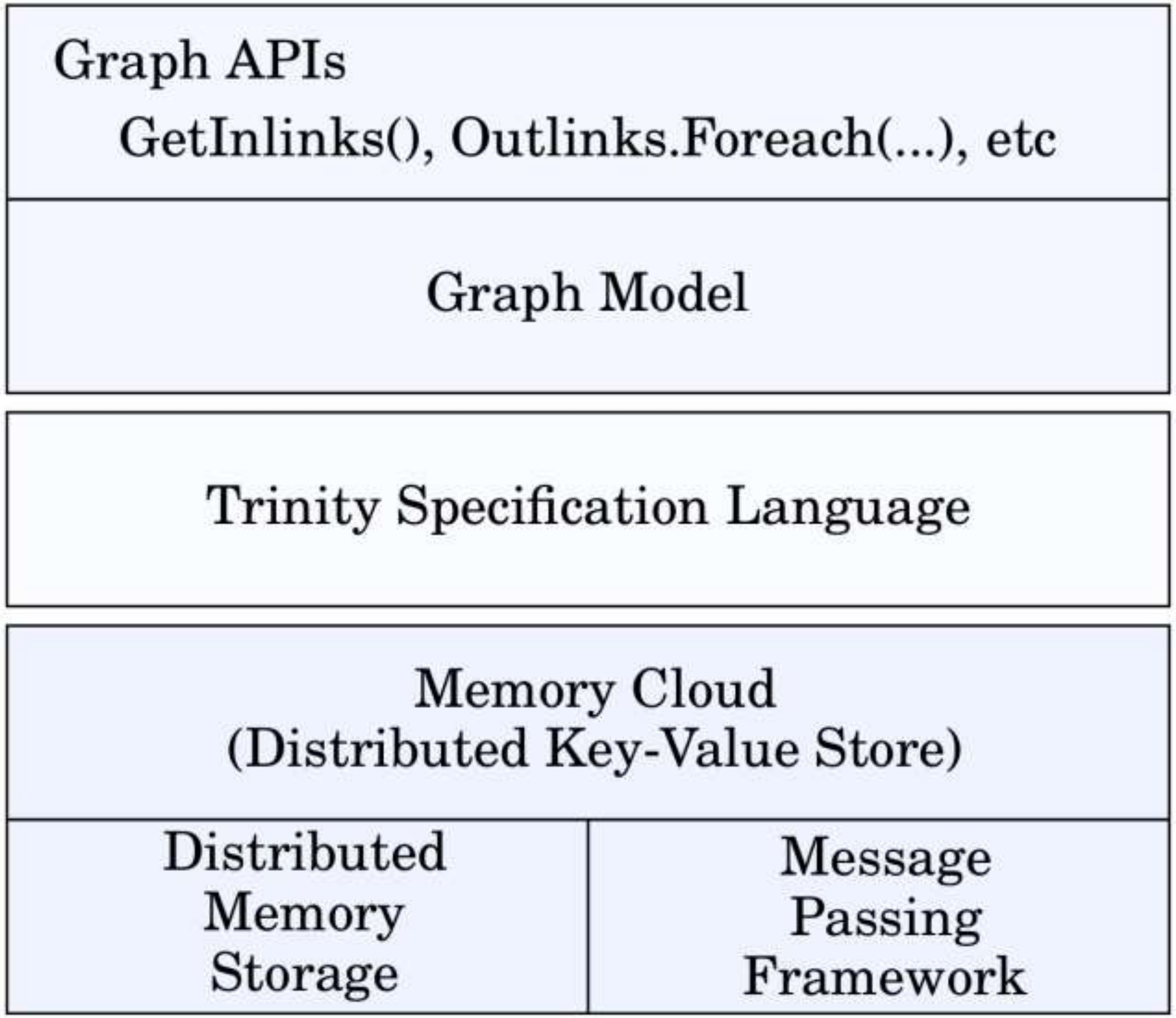
Fast random access

Parallel computation

Low latency online query processing

High throughput offline analytics

System Stack



Graph APIs

GetInlinks(), Outlinks.Foreach(...), etc

Graph Model

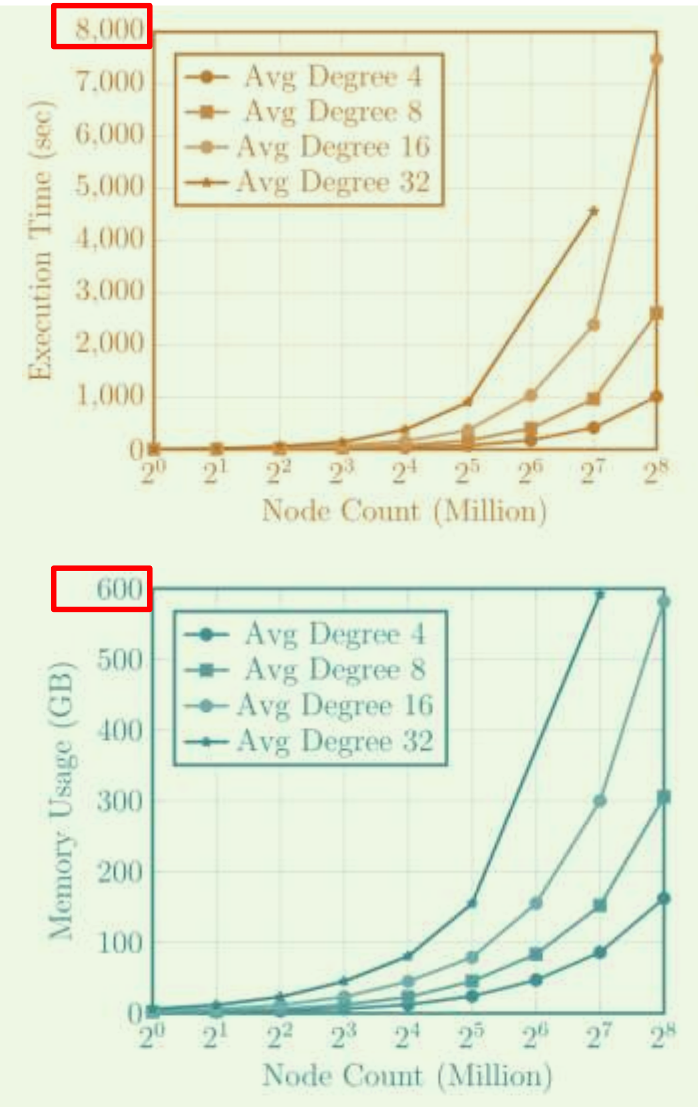
Trinity Specification Language

Memory Cloud
(Distributed Key-Value Store)

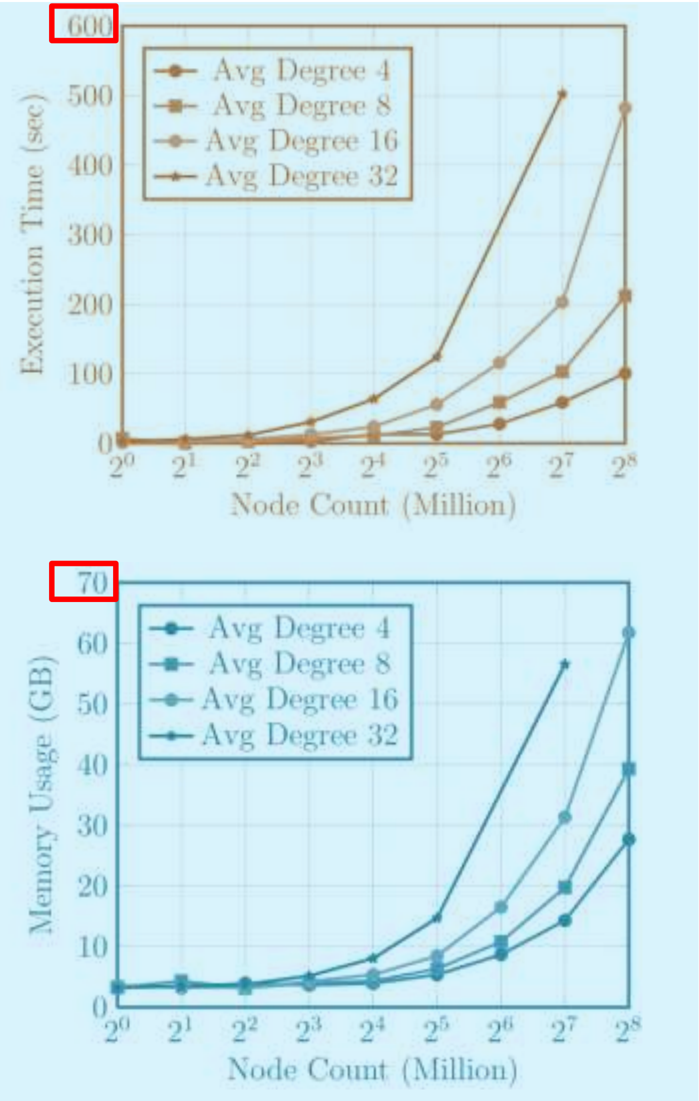
Distributed Memory Storage

Message Passing Framework

One Byte Counts (Trinity vs. PBGL)



PGBL

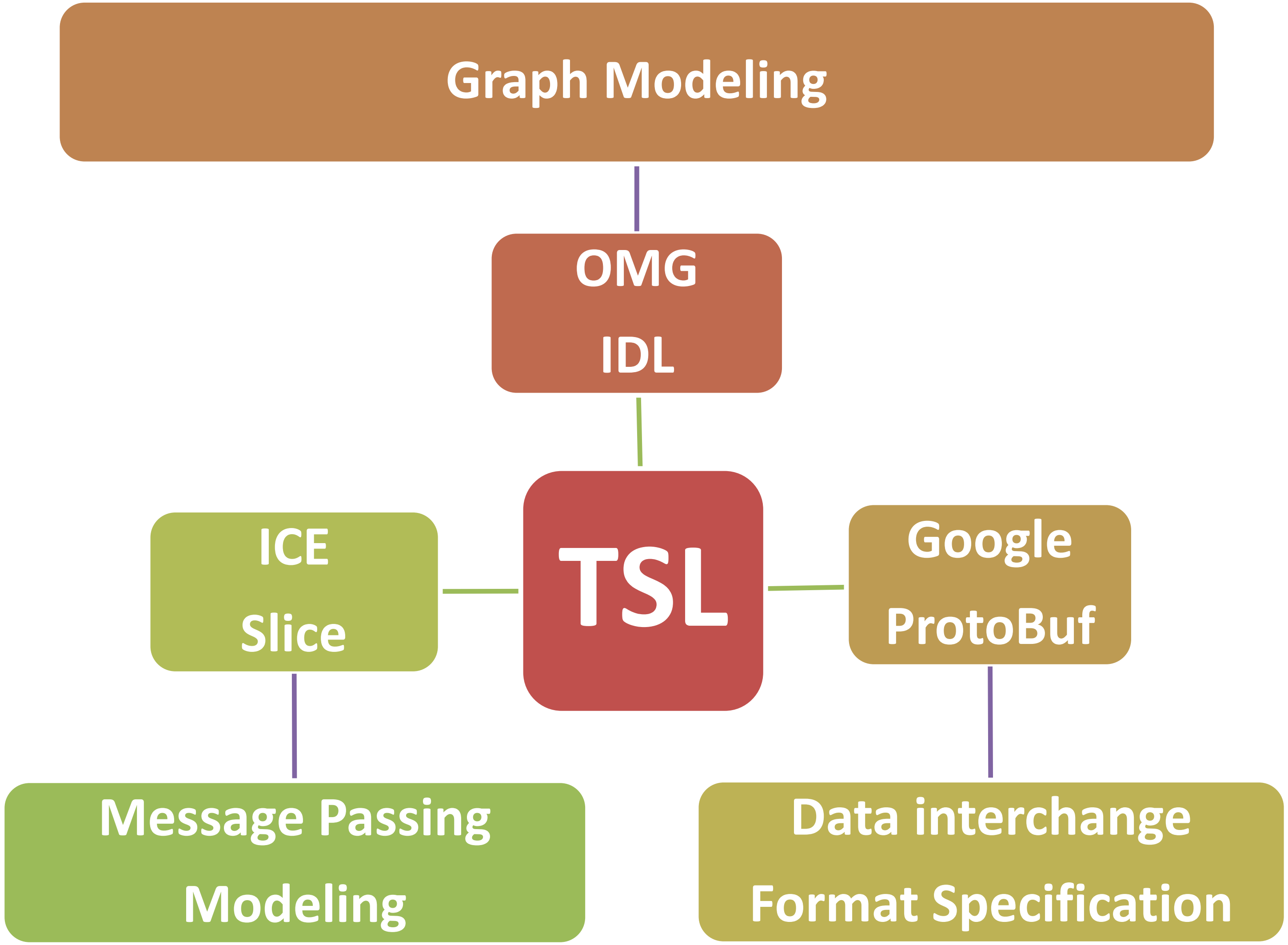


Trinity

Execution Time

Memory Usage

Trinity Specification Language



Graph Modeling

OMG IDL

ICE Slice

TSL

Google ProtoBuf

Message Passing Modeling

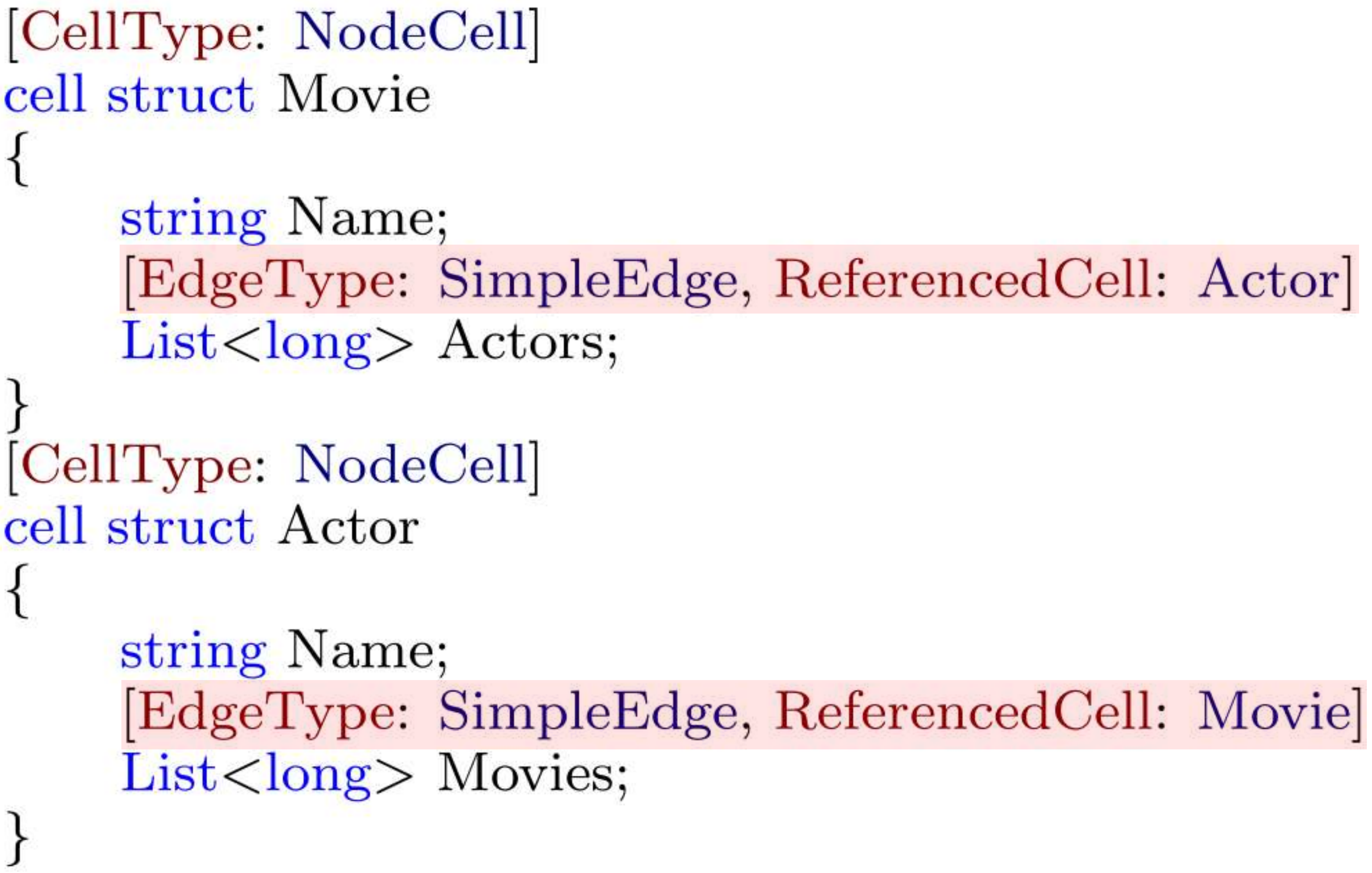
Data interchange Format Specification

Why TSL?

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

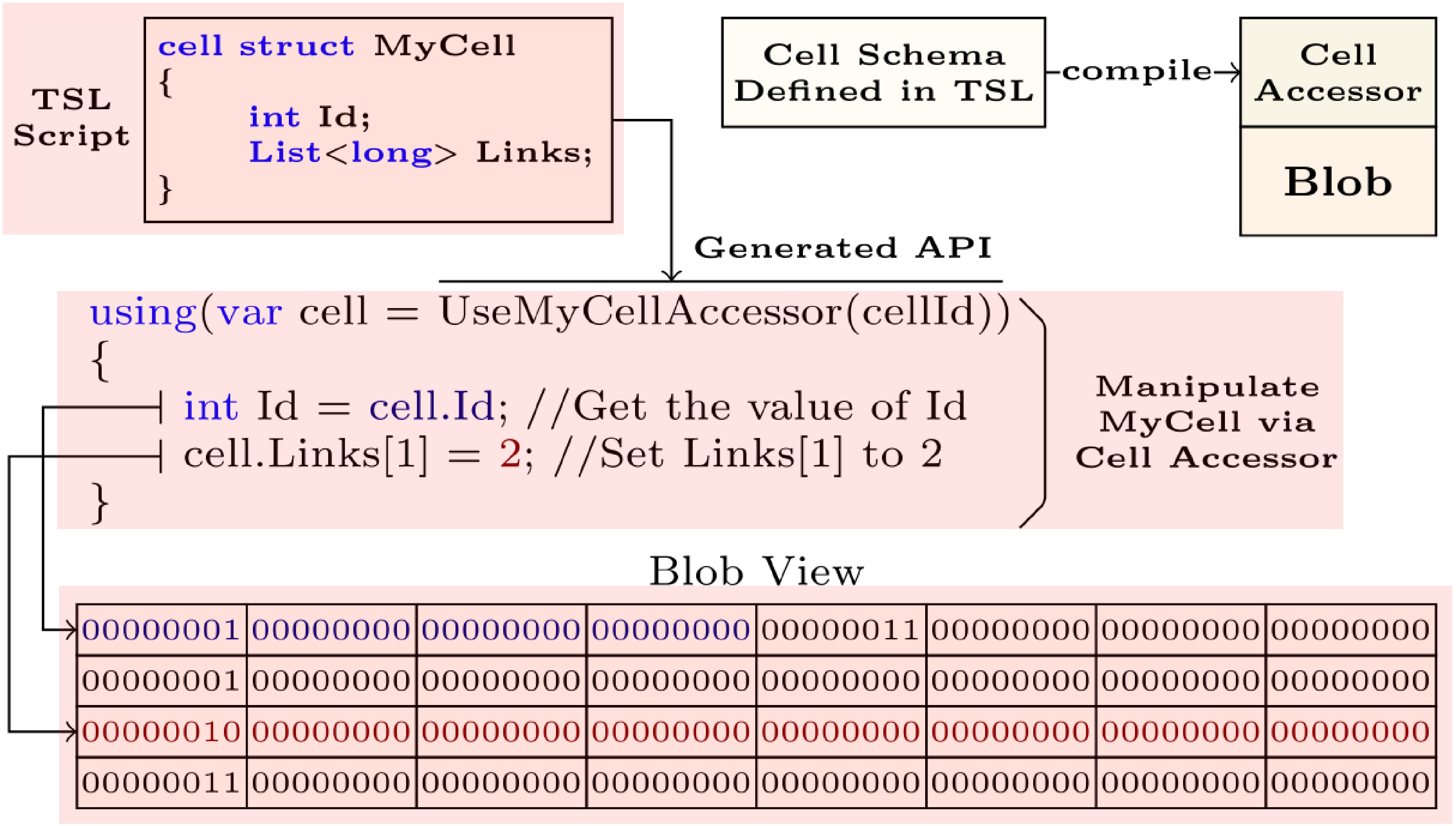
- TSL makes Trinity memory cloud beyond a key-value store
  - Users are allowed to freely define the data schema
  - TSL makes message passing programming ever so easy

Modeling a Movie and Actor Graph



```
[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

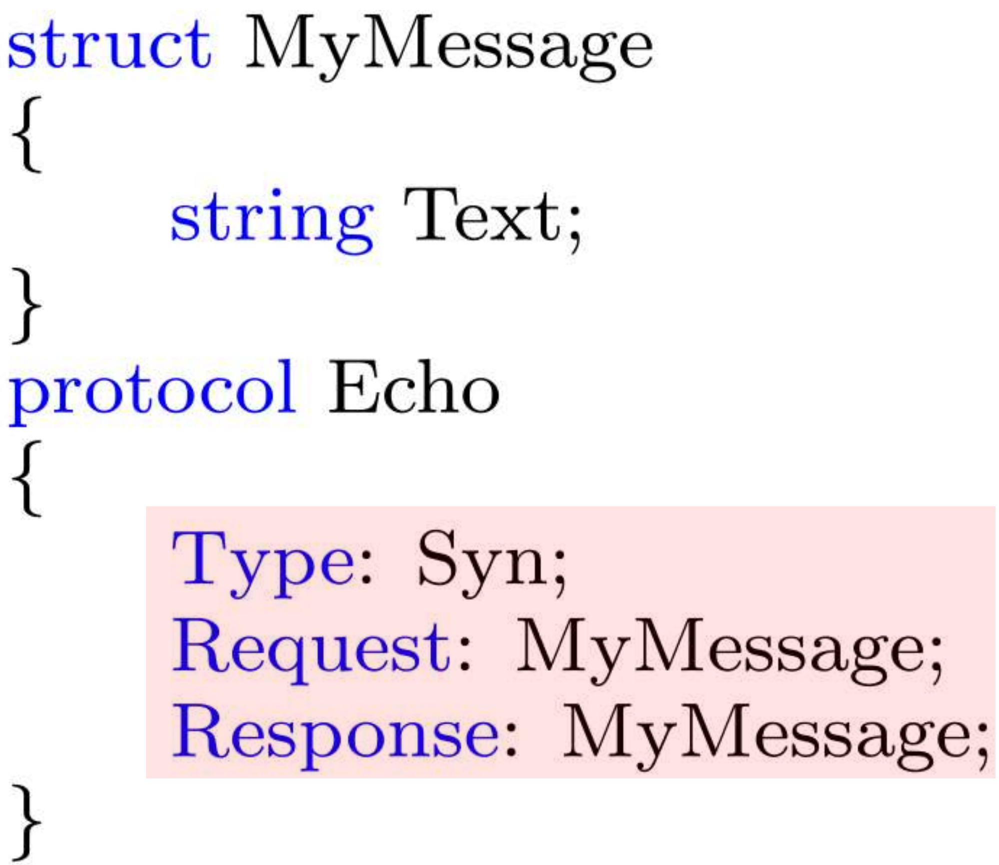


```
cell struct MyCell
{
      int Id;
      List<long> Links;
}
```

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

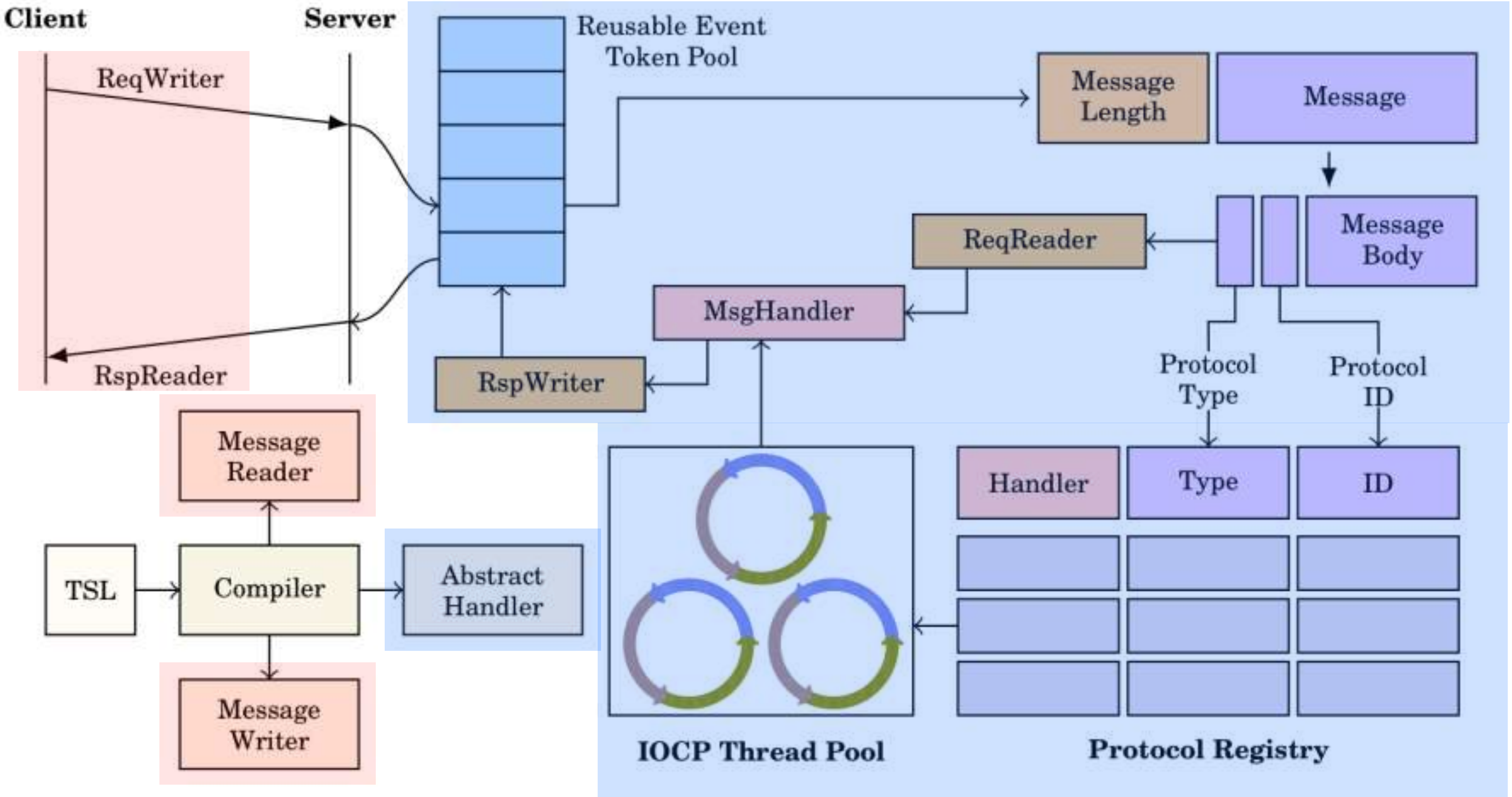
00000001	00000000	00000000	00000000	00000011	00000000	00000000	00000000
00000001	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000010	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000011	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Modeling Message Passing



```
struct MyMessage
{
        string Text;
}
protocol Echo
{
        Type: Syn;
        Request: MyMessage;
        Response: MyMessage;
}
```

TSL-Powered Message Passing



Trinity-enabled Graph Computation Paradigms

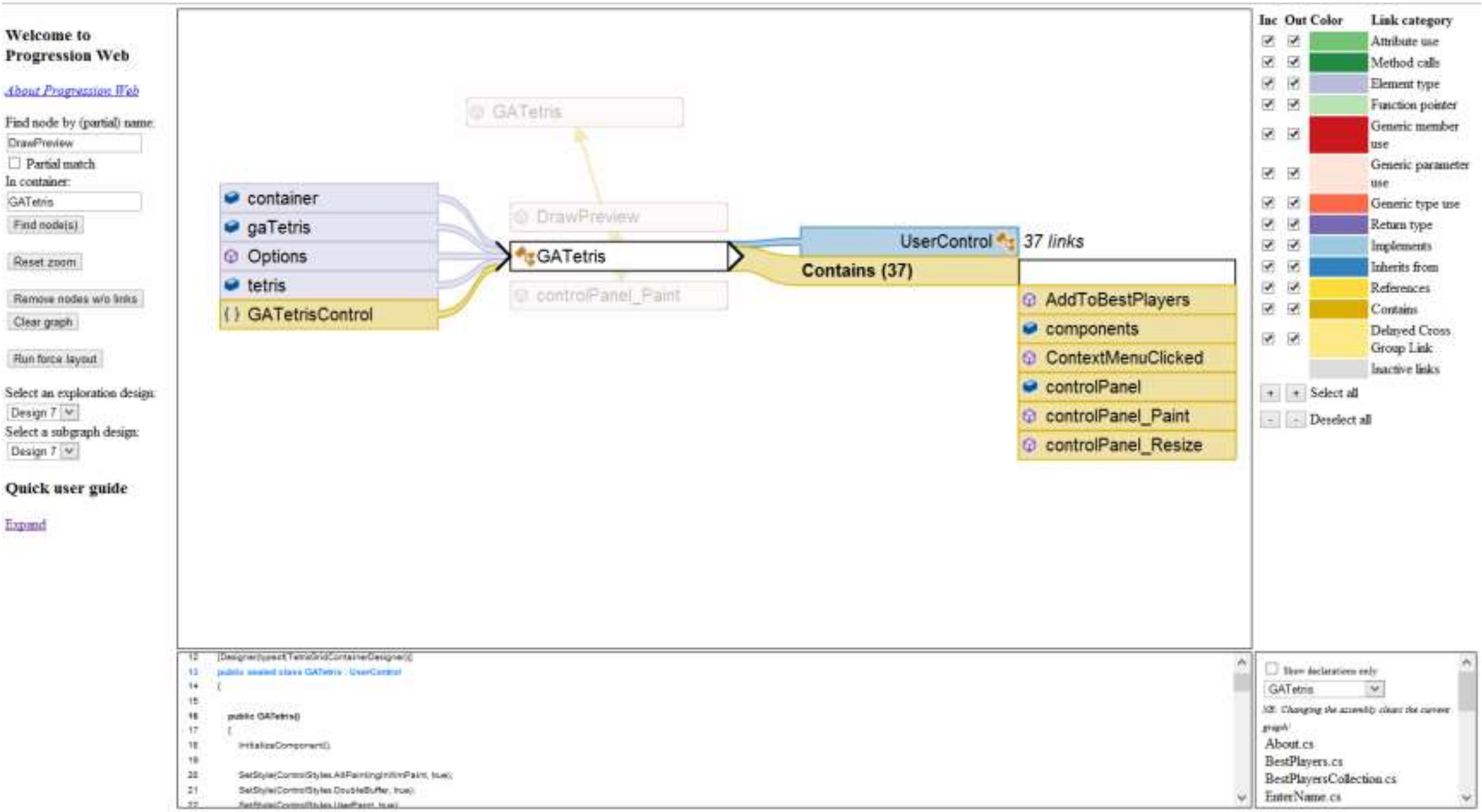
- **Vertex-centric graph analytics**
  - Prosperous since Pregel, e.g. Giraph, GraphChi

- **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)

Trinity Applications

Source Code Graph (Visual Studio)



Welcome to Progression Web

About Progression Web

Find node by (partial) name: DrawPreview

☐ Partial match

In container: GATetris

Find node(s)

Reset zoom

Remove nodes w/o links

Clear graph

Run force layout

Select an exploration design: Design 7

Select a subgraph design: Design 7

Quick user guide

Expand

Inc	Out	Color	Link category
☑	☑	🟩	Attribute use
☑	☑	🟩	Method calls
☑	☑	🟪	Element type
☑	☑	🟩	Function pointer
☑	☑	🟥	Generic member use
☑	☑	🟧	Generic parameter use
☑	☑	🟧	Generic type use
☑	☑	🟪	Return type
☑	☑	🟦	Implements
☑	☑	🟦	Inherits from
☑	☑	🟨	References
☑	☑	🟨	Contains
☑	☑	🟨	Delayed Cross Group Link
		⬜	Inactive links
+	+		Select all
-	-		Deselect all

```
[Designer(typeof(TetrisGridContainerDesigner))]
public sealed class GATetris : UserControl
{

    public GATetris()
    {
        InitializeComponent();

        SetStyle(ControlStyles.AllPaintingInWmPaint, true);
        SetStyle(ControlStyles.DoubleBuffer, true);
        SetStyle(ControlStyles.UserPaint, true);
```

☐ Show declarations only

GATetris

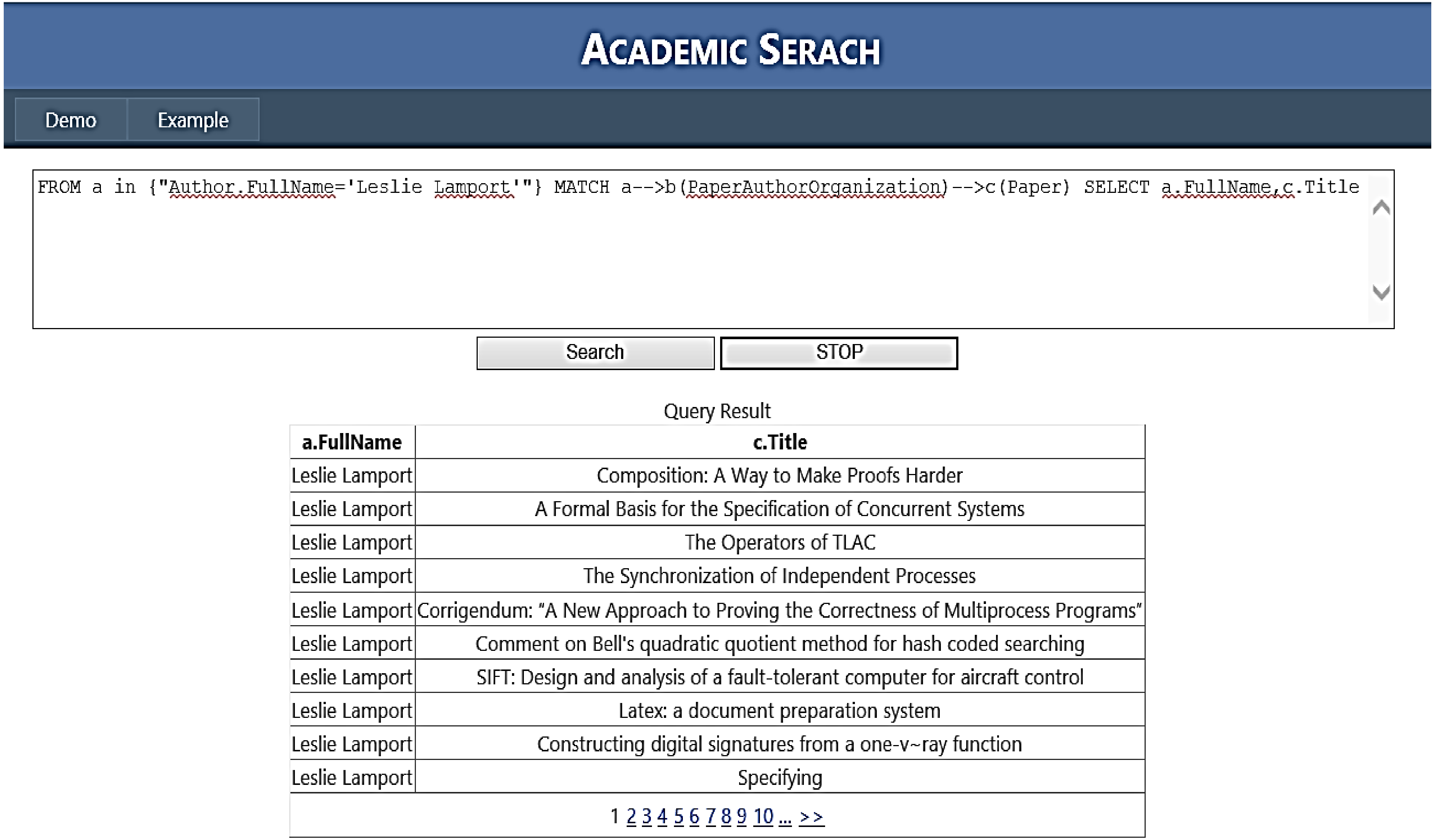
NB: Changing the assembly clears the current graph!

About.cs

BestPlayers.cs

BestPlayersCollection.cs

EnterName.cs

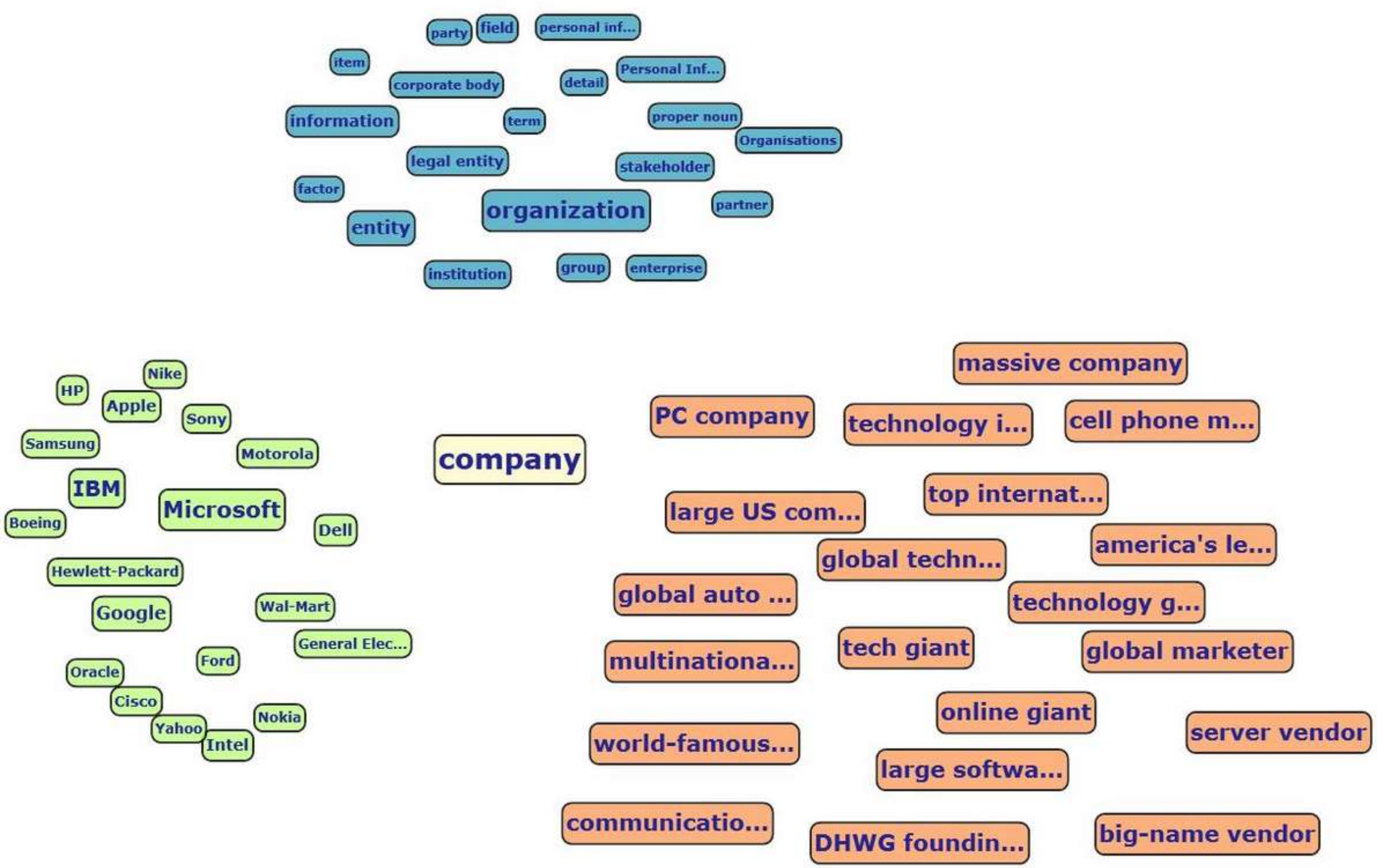


```
FROM a in {"Author.FullName='Leslie Lamport'"} MATCH a-->b(PaperAuthorOrganization)-->c(Paper) SELECT a.FullName,c.Title
```

Query Result

a.FullName	c.Title
Leslie Lamport	Composition: A Way to Make Proofs Harder
Leslie Lamport	A Formal Basis for the Specification of Concurrent Systems
Leslie Lamport	The Operators of TLAC
Leslie Lamport	The Synchronization of Independent Processes
Leslie Lamport	Corrigendum: "A New Approach to Proving the Correctness of Multiprocess Programs"
Leslie Lamport	Comment on Bell's quadratic quotient method for hash coded searching
Leslie Lamport	SIFT: Design and analysis of a fault-tolerant computer for aircraft control
Leslie Lamport	Latex: a document preparation system
Leslie Lamport	Constructing digital signatures from a one-v~ray function
Leslie Lamport	Specifying

Knowledge Graph



Thanks!

http://research.microsoft.com/trinity/