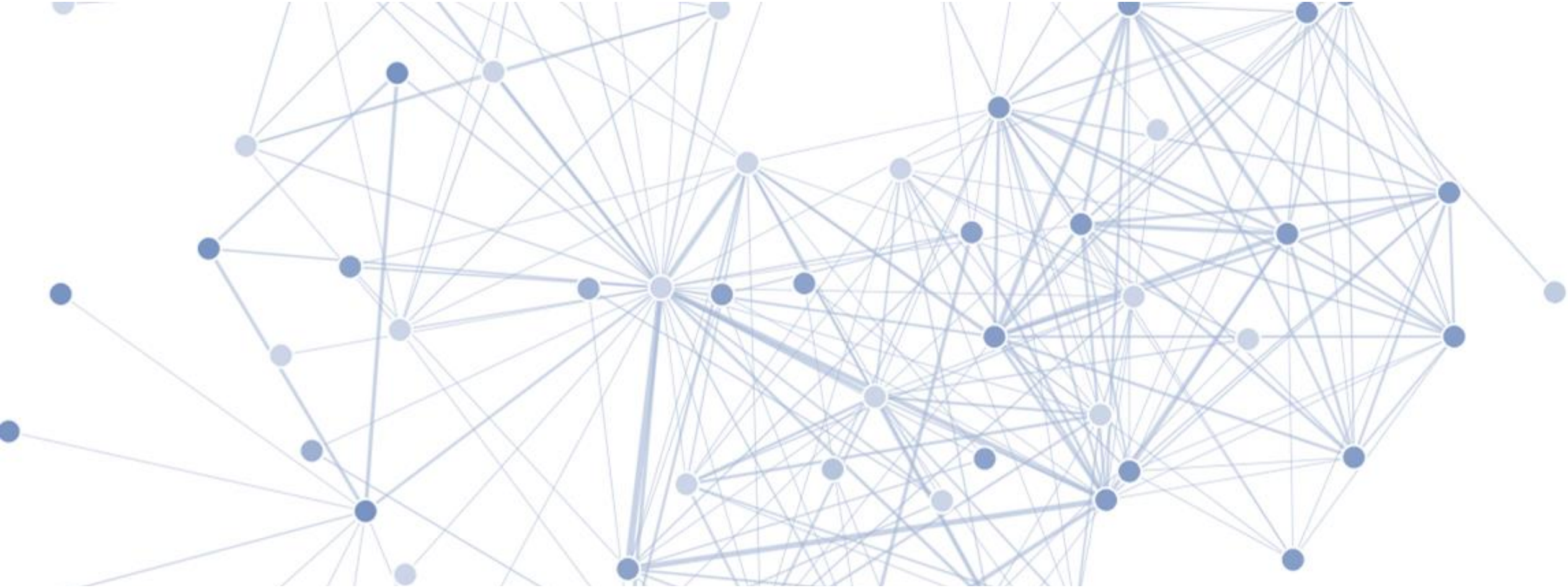


A Comparison of MapReduce Join Algorithms for RDF

Albert Haque[1] and David Alves[2]
Research in Bioinformatics and Semantic Web Lab, University of Texas at Austin
[1]Department of Computer Science, [2]Department of Electrical and Computer Engineering
April 18, 2014

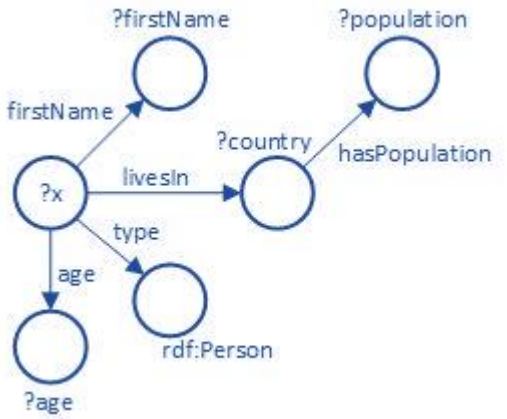
Background: Cloud Triple Store

- Property table
  - Subject as row key
  - Predicates (objects) as columns
  - Objects (predicates) as values

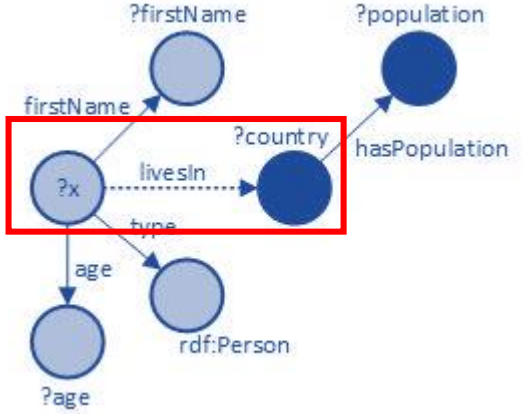
```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?firstName, ?age, ?population
WHERE {
            ?x rdfs:firstName ?firstName .
            ?x rdfs:age ?age .
            ?x rdf:type rdf:Person .
            ?x rdfs:livesIn ?country .
            ?country rdf:population ?population
}
```

- Need to join two subjects

SPARQL Query



Task of SPARQL to Hive Transformation



Algorithms

1. Map Side Join
2. Semi Join (Map Side Join)
3. Merge-Sort Join (Reduce Side Join)
4. Repartition Join (Reduce Side Join)

Map-Side Join

Pre-Map Phase [1]

1. Each table is split into same number of partitions
2. Sort each table by the join key
3. All records for a particular key must be in same partition
   - Typically achieved by running a Reduce job beforehand

Map Phase

- For each partition, scan the data and join data based on key
- Emit resulting tuple

Reduce Side

- None

[1] White, T. *Hadoop: The Definitive Guide, 2nd Edition*. O'Reilly Media/Yahoo Press. September 2010.

Semi-Join

- When ***R*** is large, there are many records in ***R*** that may not be referenced in ***S*** (assuming $R \bowtie S$)
- Semi-join dramatically reduces the data sent over network
- Requires 3 MapReduce jobs

  Job 1: Get a list of unique join keys, ***S.uk***. (Map+Reduce)

  Job 2: Load ***S.uk*** into memory and loop through ***R***. If a record's key is found in ***S.uk***, emit it. Now we have a list of records in ***R*** to be joined.

  Job 3: Use a broadcast join to perform the join with ***S***

Broadcast Join: If $|R| \ll |S|$ then send ***R*** to all mapper nodes [1]

[1] Blanas, S., et al. A comparison of Join Algorithms for Log Processing in MapReduce. SIGMOD 2010.

Sort-Merge Join

- Assume $R \bowtie S$
- Map Phase:
  - Key ***k***: Both datasets ***R*** and ***S*** have their join attributes extracted
  - Value ***v***: Rows from ***R*** or ***S*** with join attribute = ***k***
  - Tag ***t***: Identifies which table ***(k,v)*** belongs to
- Reduce Phase:

  for each $\boldsymbol{k_1}$ with ***t=R***:

  for each $\boldsymbol{k_2}$ *with* ***t=S****:*

  if $\boldsymbol{k_1 = k_2}$:

  $emit(\boldsymbol{k_1}, \boldsymbol{v_1}, \boldsymbol{v_2})$

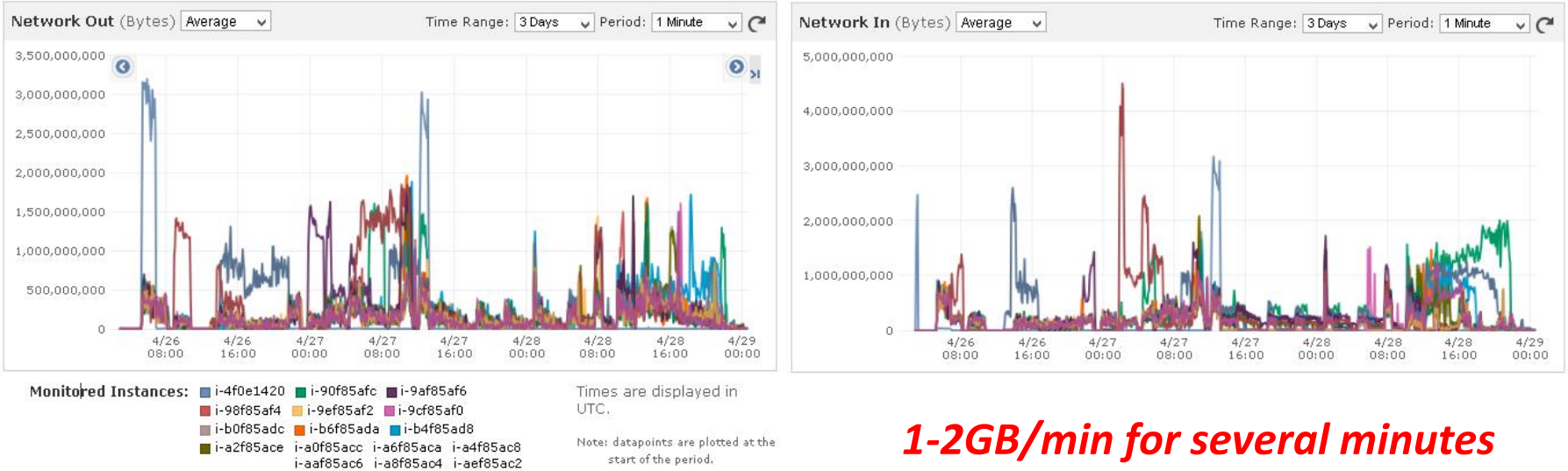
Sort-Merge Join

- Pros:
  - Easiest to implement

Problems:

1. Data is scattered across HDFS
2. Sends entire dataset across network

Hadoop, HBase, & Hive – BSBM 1 billion triples, 16 nodes on Amazon EC2 [1,2]



1-2GB/min for several minutes

[1] A. Haque. "A MapReduce Approach to NoSQL RDF Databases". The University of Texas at Austin, Department of Computer Science(1). Report# HR-13-13 (honors theses). December 2013.

[2] P. Cudré-Mauroux, I. Enchev, S. Fundatureanu, P. Groth, A. Haque, A. Harth, F. Keppmann, D. Miranker, J. Sequeda, and M. Wylot. *NoSQL Databases for RDF: An Empirical Evaluation.* Proceedings of the 12th International Semantic Web Conference (ISWC). Oct 2013.

Repartition Join

- Uses Compound Keys: **tag+key** [1]
  - Where **tag** is an identifier for the parent table

(key,value) = (tag+key, value) = (t1|subject, <kv1,kv2,…,kvn>)

- Partition phase hashes the key part of the compound key
  - Guarantees tuples with same join key are sent to same reducer
- Intermediate data is sorted only by key part
- We load the smaller relation into memory by using the tag portion of compound key and perform the join

[1] Blanas, Spyros, et al. "A comparison of join algorithms for log processing in mapreduce." *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010.

Data Model

- Subjects as row keys
- Objects as columns
- Predicates as cell values
- Why? Because we use bloom filters

- Note: literal types such as “price” have the predicate as the column and object as the cell value

Predicate Literals | **Non-Literal Objects**

Subjects

Row Key	price	date	label	…	product1	product2	feature29	feature502	Mexico	…
product1	1099.99		MacBook Air	…			hasFeature			…
…				…						…
product1023	479.99		Surface Pro	…			hasFeature	hasFeature		…
offer1		1984930000		…	offerFor				originatedFrom	…
offer9230		1239840000		…		offerFor				…
producer482			Fabricación de Ordenadores	…	producerOf		specializesIn		hasFactoryIn	…
review32901		1093920330	Very thin, works great!	…	reviewOf					…

Object Literals

Non-Literal Predicates

Datasets

1. Berlin SPARQL Benchmark (BSBM)

Dataset Size	Exact Number of Triples	Table Rows (Unique Subjects)	File Size (.nt)	Scale Factor (-pc)
10 million	10,065,245	934,324	2.5 GB	29,000
100 million	100,652,457	9,197,305	25.0 GB	290,000
1 billion	1,004,406,629	*Estimated* 91,973,050	250.0 GB	2,900,000
10 billion	*Estimated* 10,044,066,290	*Estimated* 919,730,500	2.5 TB	29,000,000

2. Leigh University Benchmark (LUBM)

Dataset Size	Exact Number of Triples	Table Rows (Unique Subjects)	File Size (.nt)	Number of Universities
10 million	11,063,815	1,744,926	1.8 GB	80
100 million	110,127,934	17,358,066	18.0 GB	800
1 billion	1,026,612,765	*Estimated* 173,580,660	183.0 GB	8,000
10 billion	*Estimated* 10,266,127,650	*Estimated* 1,735,806,600	1.83 TB	80,000

Not All Benchmark Queries Were Used

- Primary purpose of this project is to examine join algorithms
- To save time and cost, we excluded all algorithms with no joins and those with simple query paths

We used six queries from each dataset:

- Berlin SPARQL Benchmark: Queries 2, 5, 7, 8, 10, 12
- Leigh University Benchmark: Queries 2, 7, 8, 9, 10, 12

Hardware Configuration

- Amazon EC2 and Elastic MapReduce
- 1, 2, 4, 8, 16, 32*, and 64* compute nodes (m1.large) + 1 master node (m1.large)
- Hadoop 1.0.3 and HBase 0.92

Cluster Compute Nodes:

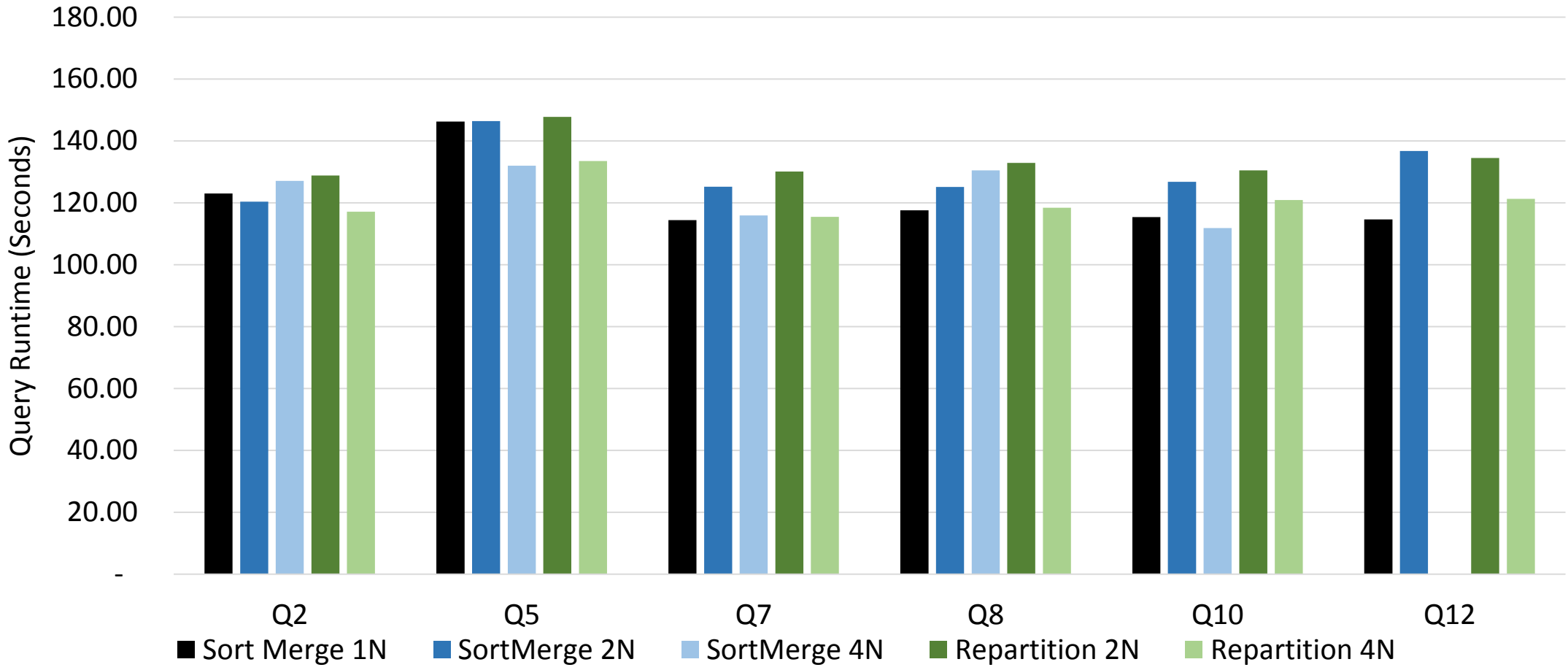
- Intel Core 2 Duo T6400 @ 2.00 GHz
- 8 GB main memory
- 840 GB (2x420) local disk storage
- Amazon 2vCPUs/4 ECUs

Dataset Generator/Preprocessing Nodes:

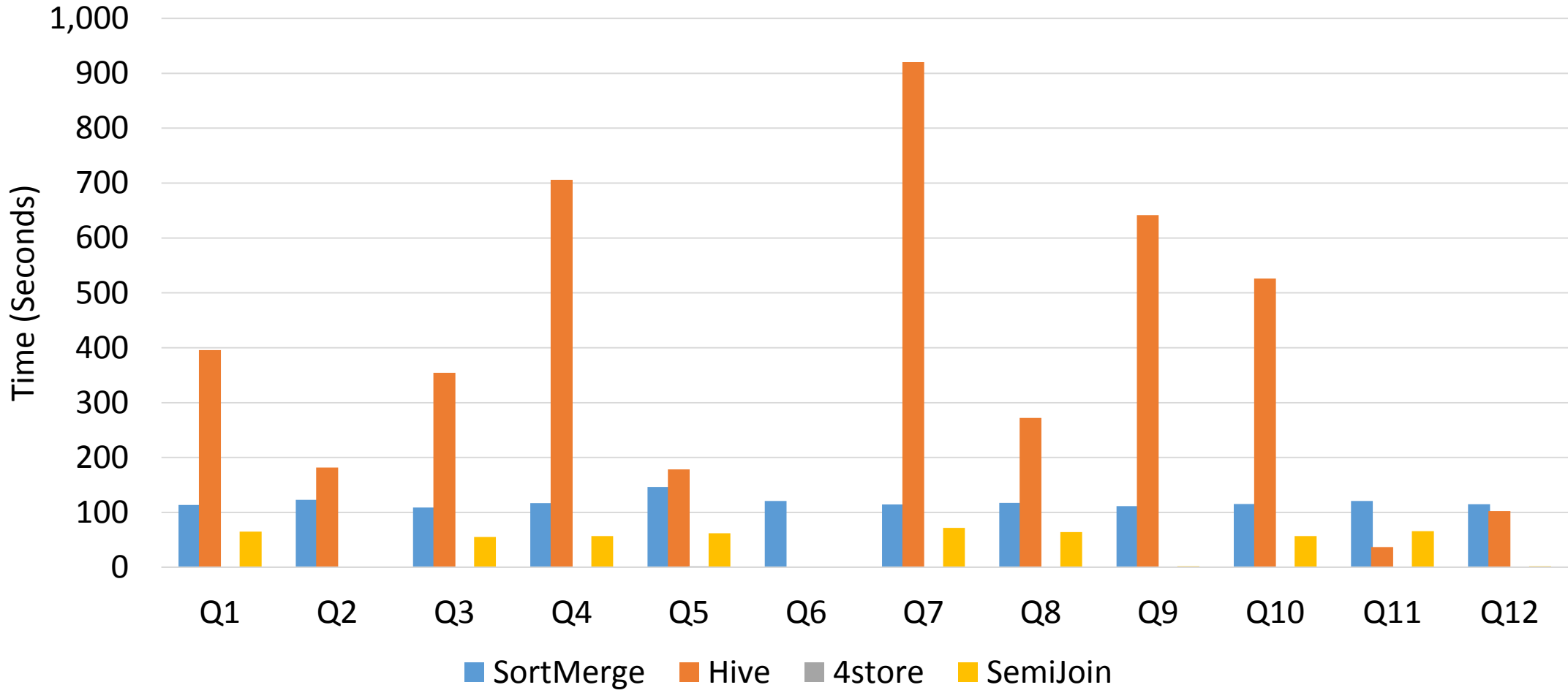
- Intel Xeon E5-2680 32-core @ 2.80 GHz
- 60 GB main memory
- 640 GB SSD (2x320) local storage
- 10 Gigabit Ethernet
- 32 Amazon vCPUs/108 ECUs

* Not used for datasets less than 1 billion triples or less than 100 GB

Results: BSBM 10 Million Triples



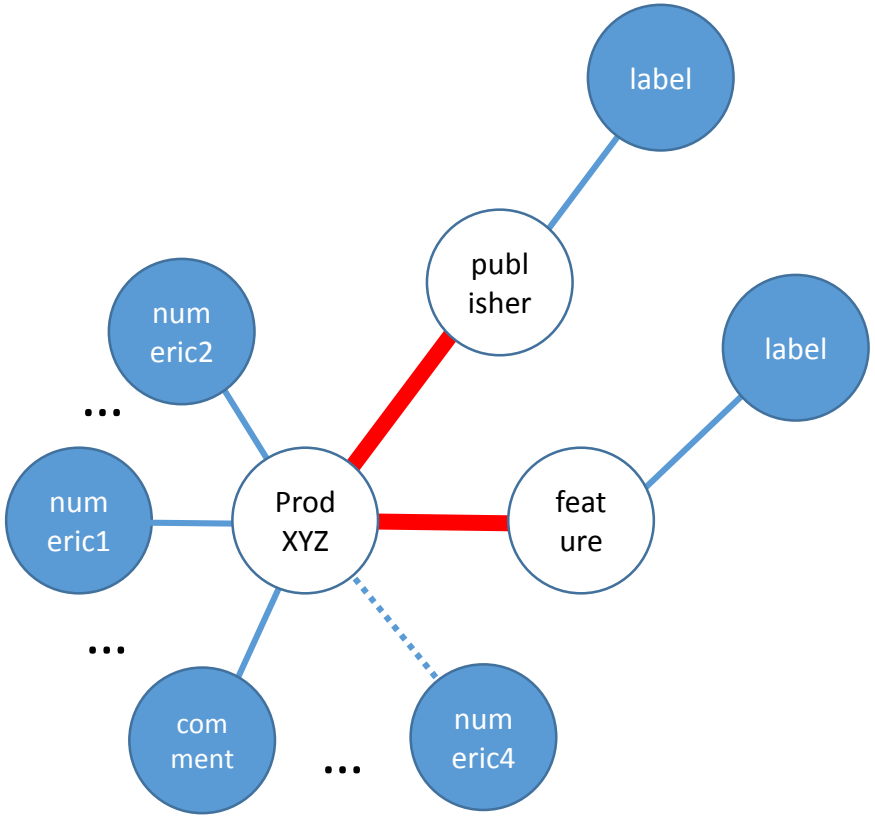
BSBM 10 million on 1 Node



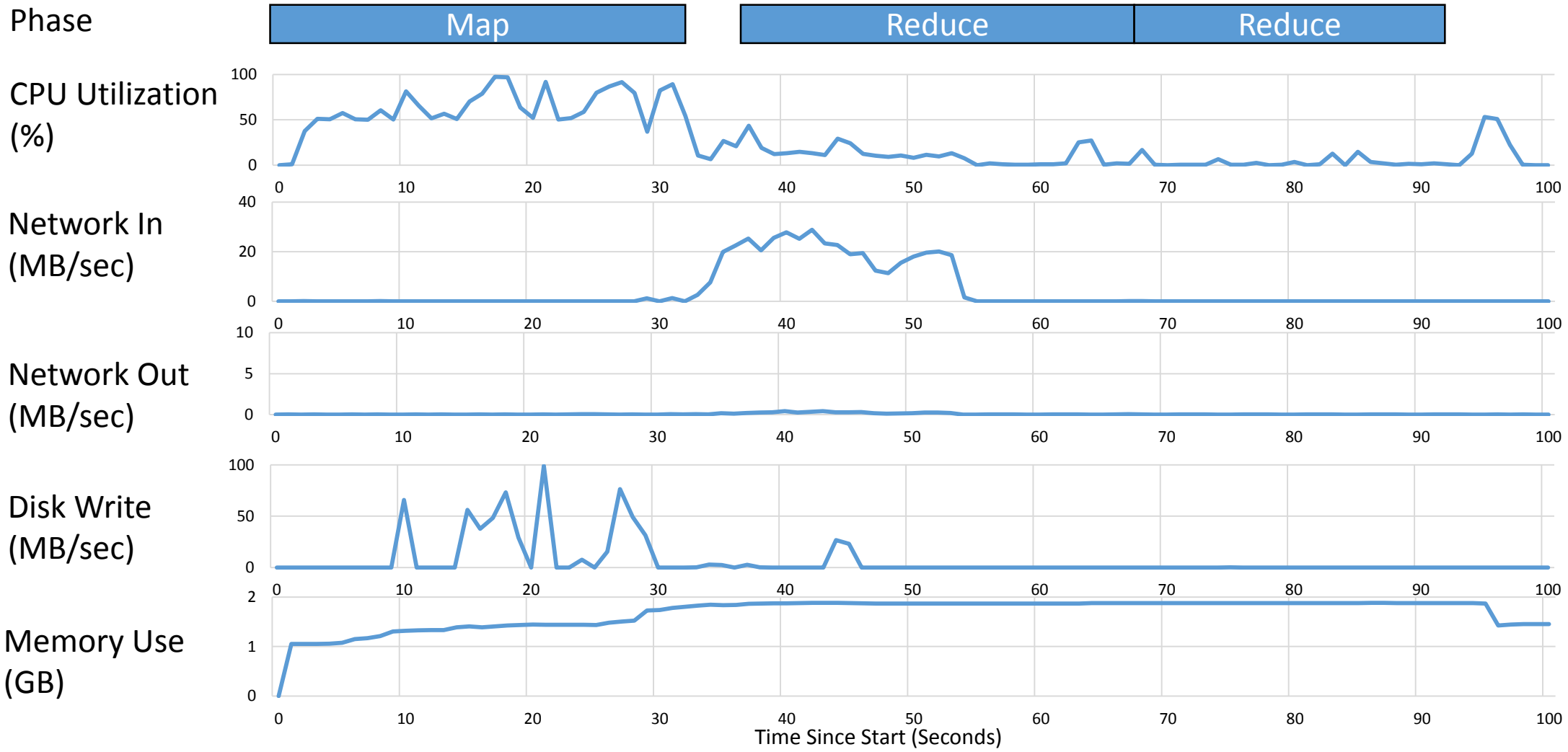
BSBM Query 2

- We analyzed this query in more detail

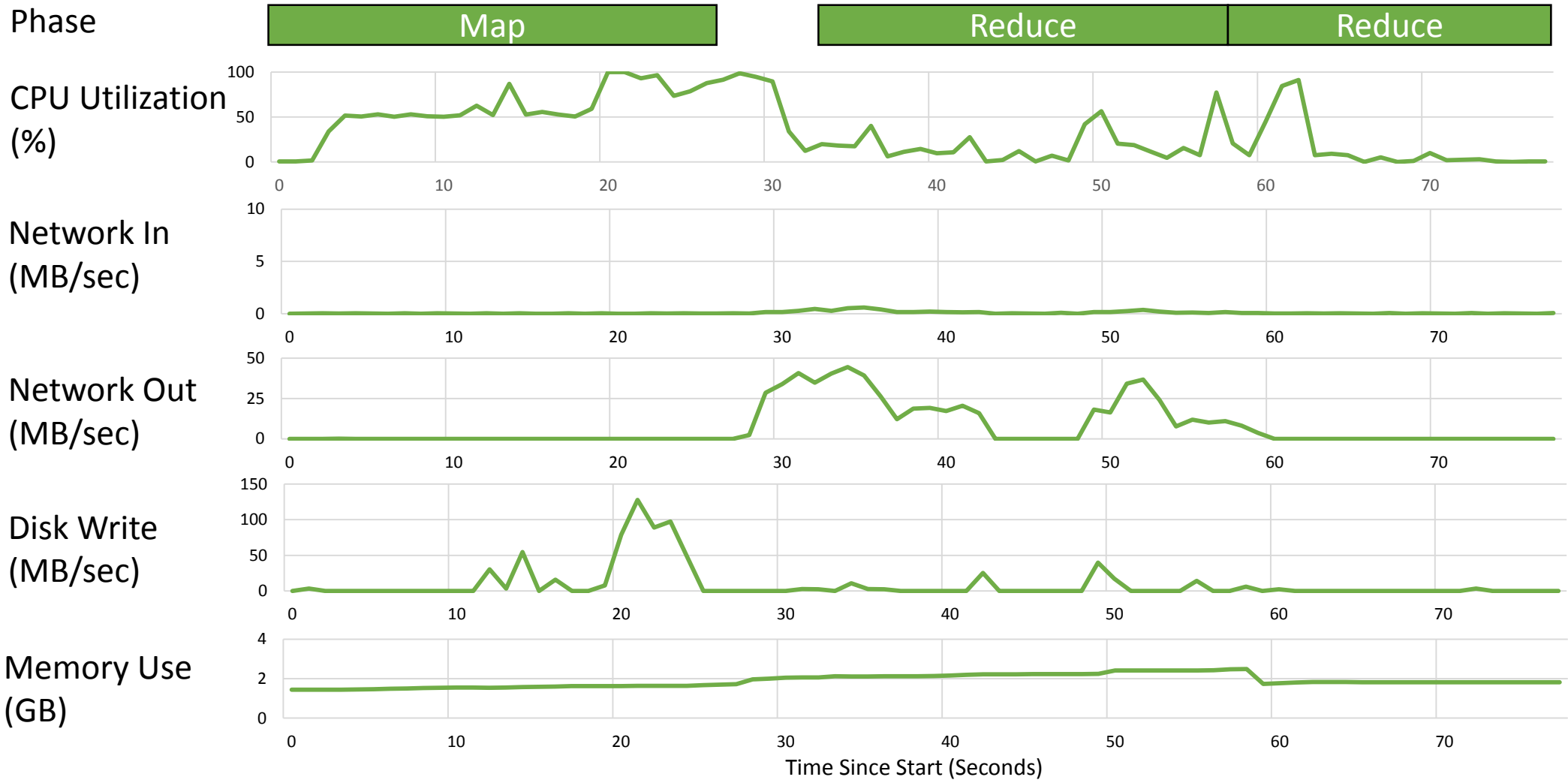
```
SELECT ?label ?comment ?producer ?productFeature ?propertyTextual1 ?propertyTextual2
?propertyTextual3 ?propertyNumeric1 ?propertyNumeric2 ?propertyTextual4
?propertyTextual5 ?propertyNumeric4
WHERE {
            %ProductXYZ% rdfs:label ?label .
            %ProductXYZ% rdfs:comment ?comment .
            %ProductXYZ% bsbm:producer ?p .
            ?p rdfs:label ?producer .
            %ProductXYZ% dc:publisher ?p .
            %ProductXYZ% bsbm:productFeature ?f .
            ?f rdfs:label ?productFeature .
            %ProductXYZ% bsbm:productPropertyTextual1 ?propertyTextual1 .
            %ProductXYZ% bsbm:productPropertyTextual2 ?propertyTextual2 .
            %ProductXYZ% bsbm:productPropertyTextual3 ?propertyTextual3 .
            %ProductXYZ% bsbm:productPropertyNumeric1 ?propertyNumeric1 .
            %ProductXYZ% bsbm:productPropertyNumeric2 ?propertyNumeric2 .
      OPTIONAL { %ProductXYZ% bsbm:productPropertyTextual4 ?propertyTextual4 }
      OPTIONAL { %ProductXYZ% bsbm:productPropertyTextual5 ?propertyTextual5 }
      OPTIONAL { %ProductXYZ% bsbm:productPropertyNumeric4 ?propertyNumeric4 }
}
```



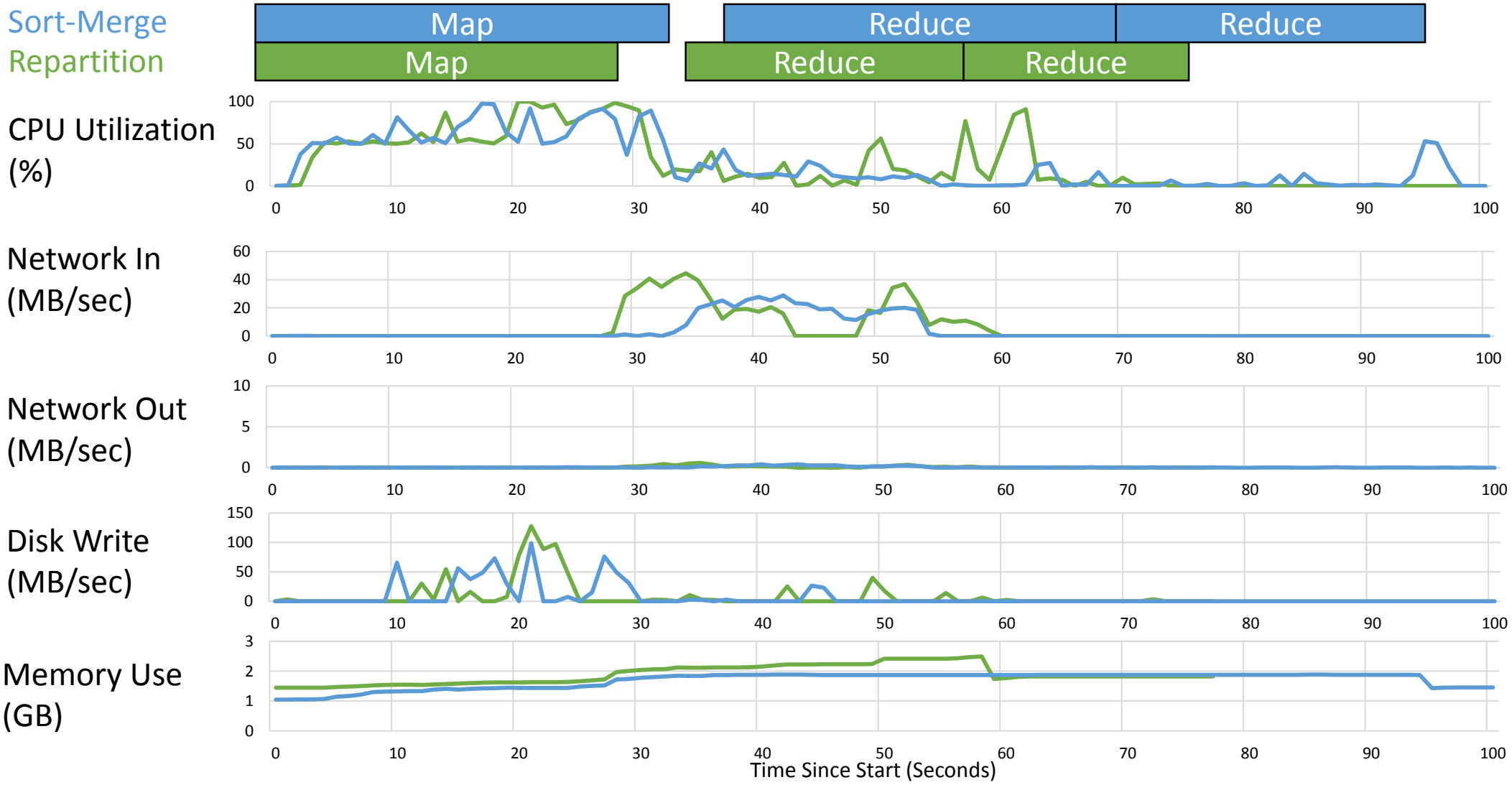
System Resources: Sort-Merge Join



System Resources: Repartition Join

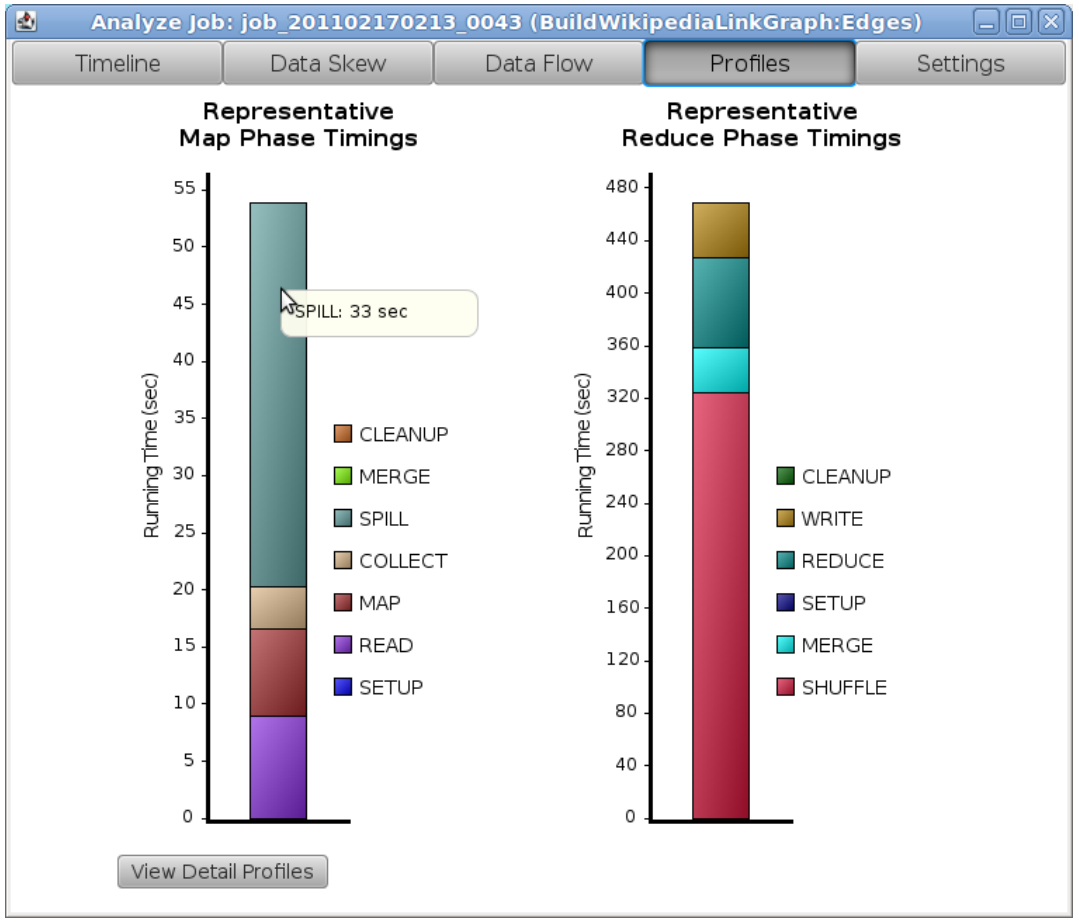


System Resources: Comparison



Currently Underway

- Designing table schema/loading technique for Leigh University Benchmark
- Query characteristics/selectivity
- Starfish – Hadoop Profiler [1]
  - Detailed analysis of Map and Reduce phases
  - Free!



[1] https://www.cs.duke.edu/starfish/