The Lusail System: Extra Supported Features and Sample Queries

Ibrahim Abdelaziz *, Essam Mansour ◇, Mourad Ouzzani ◇, Ashraf Aboulnaga ◇, Panos Kalnis *

* *King Abdullah University of Science & Technology, Saudi Arabia*
{first}.{last}@kaust.edu.sa

◇ *Qatar Computing Research Institute, HBKU, Qatar*
{emansour,mouzzani,aaboulnaga}@qf.org.qa

Lusail is a system for scalable and efficient SPARQL query processing over decentralized graphs. This document discusses the extra supported features by Lusail, such as multiple query optimizations (Section I) and the ability to utilize multiple machines for query execution (Section I). Furthermore, in Section III we analyze the effect of the different query optimizations proposed by Lusail. Finally, we show sample queries from each benchmark (Sections IV-VIII).

I. Multiple-Query Optimization

Lusail allows multi-query optimization (MQO) through a user-defined time window on which the system continues to receive queries and evaluate them together as a single workload. For a batch of N queries, Lusail analyzes each query individually and decomposes it into a set of subqueries (see Section IV). Then, Lusail detects the common subqueries among all sets of subqueries. In different queries, if two or more subqueries share the same predicates and the same number of variables, then they form a common subquery. Lusail maintains the list of common subqueries across the N queries. Then, Lusail estimates the cost of these subqueries and decides delayed and non-delayed subqueries. Notice that, the delay decision tend to be more accurate as the number of subqueries increases which ensures larger population and better estimation of the mean and standard deviation.

In MQO mode, Lusail first evaluates the common non-delayed subqueries. Then, it iterates over all queries to produce the final result. There are three possibilities for the query evaluation: (*i*) when the query contains only non-delayed subqueries, Lusail joins their computed results and reports the final answer. (*ii*) When the query contains both delayed and non-delayed subqueries, Lusail uses its selectivity-aware technique (SAPE) that utilizes bound joins for evaluating delayed subqueries. Finally, (*iii*) if all the subqueries are delayed, Lusail detects the subquery with the minimal cost, change its status to non-delayed and utilizes SAPE to find the query results.

We conduct an experiment to evaluate the effectiveness of our MQO add-on feature. We use the LUBM workload in [2]. This workload consists of 10K unique queries which are derived from the 14 LUBM benchmarks by changing their structures and constants. We generate workloads of different sizes by randomly choosing queries from the pool of the 10K unique queries. We generated workloads of 10, 20, 40, 80 and 160 queries. Figure 1 shows the effect of the MQO of Lusail compared to evaluating queries independently. As the workload size increases, the possibility to find similar substructures between queries increases. Therefore, utilizing the shared computation among queries results in performance gains that range from 34% to 46%.

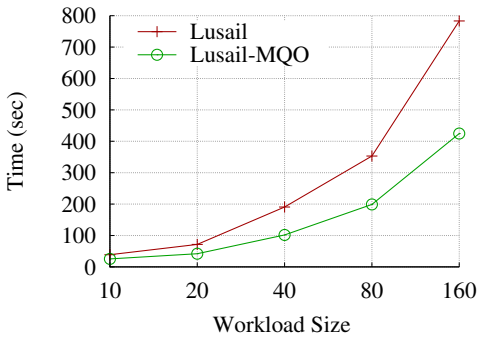


Fig. 1. Effect of MQO for different LUBM workload sizes evaluated on 256 university endpoints.

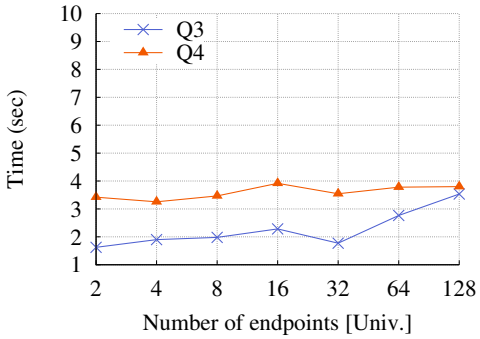
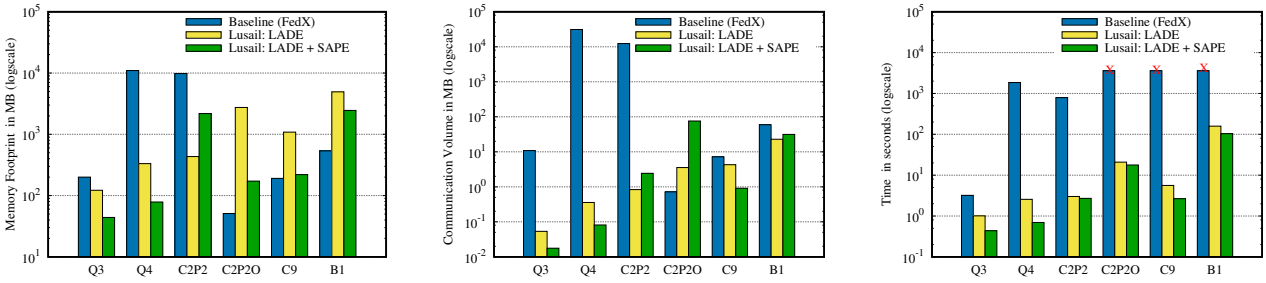


Fig. 2. Multi-machine Execution: utilizing multiple machines for query evaluation.

II. Multi-machine Execution

Recall that, Lusail creates one computing thread per endpoint. Hence, when handling a large number of endpoints, utilizing a single machine for query evaluation can be a bottleneck. We can run Lusail on multiple machines, where one of the machines will act as a master, and others will act only as workers. A worker is a Lusail instance that is assigned a number of endpoints less than or equal the number



(a) Memory Footprint (b) Communication Volume (c) Response Time

Fig. 3. Lusail optimizations can move computations on intermediate data to the endpoints, retrieve less irrelevant data, and achieve orders of magnitudes speedup compared to FedX. X corresponds to time out.

of cores in the instance. One of the threads at the master is designated as a query coordinator. A query coordinator sends the subqueries and their corresponding endpoints information to the workers. Each worker evaluates its own subqueries and then collaboratively with other servers joins the partial results. The threads on all servers communicate to distribute subqueries among them and to synchronize in the join phase. At the end of the join phase, all threads communicate their results back to the query coordinator, which aggregates the results and reports them to the user.

This experiment evaluates the performance of Lusail as we scale up the amount of data and also the number of cores used by Lusail. We use the LUBM benchmark and vary the number of endpoints (universities) from 2 to 128. We run the experiments on the *480-cores* cluster. The goal of this experiment is to increase the number of cores used by Lusail and demonstrate a good scale up [1], also known as weak scalability. Specifically, we want to demonstrate that Lusail can maintain a fixed query response time as the problem size (number of endpoints) and the number of cores expand simultaneously. This type of scalability is important for federated systems, since it corresponds to being able to handle more endpoints by adding more cores, since each new endpoint comes with its own new data.

For this experiment, we deploy Lusail on 8 machines with 16 threads each. The rest of the machines are used for deploying the endpoints. We double the computing resources (cores) as we double the number of endpoints. Figure 2 shows the results using *Q3* and *Q4*, which are the non-disjoint queries. The figure shows good scale up for Lusail. The performance for *Q3* is not as good as *Q4*, since Lusail processes joins by sending the less partitioned relation (inner) to the more partitioned relation (outer) with the aim of increasing parallelism. Both *Q4* and *Q3* are decomposed into two subqueries (relations). However, the inner relation in *Q4* is very small, while the one in *Q3* is larger. Therefore, sending the inner relation in *Q3* incurs a higher communication cost.

III. Analyzing Lusail Optimizations

This experiment measures the gain obtained by the optimizations introduced by LADE and SAPE compared to FedX. We compare the two systems in terms of memory, network, and total query time. FedX and Lusail are each deployed on a single machine of the *84-cores* cluster. We only report results for two queries from each dataset; we observed the same behavior in most of the queries with medium and high complexity, see our results for QFed, LUBM, and LargeRDFBench in Sections VI.C and VI.D.

Figures 3(a), 3(b), and 3(c) show the peak memory usage, communication volume (intermediate data to be shipped), and total response time of each query, respectively. FedX takes a significant amount of time for query execution due to its static query decomposition and bound join evaluation. It could not process three queries out of six within the time limit of one hour (*C2P2O*, *C9* and *B1*). In these three queries, FedX did not consume a lot of memory or send a lot of data over the network, but rather it overwhelmed the endpoints with lots of requests and wasted the whole hour waiting for them to finish.

LADE decomposition shifts the computation on the intermediate data from Lusail to the endpoints. Therefore, for the queries that FedX was able to complete, Lusail with LADE alone (without SAPE) consumes significantly less memory and communication compared to FedX. Lusail outperforms FedX by up to three orders of magnitude in terms of query response time. Using the SAPE execution in addition to the LADE decomposition further improved the query response time of Lusail. When enabling SAPE, the delayed queries affect the memory and communication costs, sometimes positively and sometimes negatively. However, communication is performed in parallel using multiple threads. Thus, the net effect is that SAPE with LADE always improve on the query response time compared to LADE alone.

In all subsequent experiments, we report the performance of Lusail with both LADE and SAPE. All systems are allowed to cache the results of source selection. Each query is run three times and we report the average of the last two. We set a time limit of one hour per query before aborting.

IV. Bio2RDF Endpoints and Queries

The relevant endpoints to the queries R1, R2, and R3 are DrugBank[1], OMIM[2], HGNC[3], MGI[4] and PharmGKB[5].

```
PREFIX dbank: <http://bio2rdf.org/drugbank_
                              vocabulary:>
```

[1] http://drugbank.bio2rdf.org/sparql
[2] http://omim.bio2rdf.org/sparql
[3] http://hgnc.bio2rdf.org/sparql
[4] http://mgi.bio2rdf.org/sparql
[5] http://pharmgkb.bio2rdf.org/sparql

```
PREFIX mgi:<http://bio2rdf.org/mgi_vocabulary>
PREFIX hgnc:<http://bio2rdf.org/hgnc_vocabulary>
PREFIX pgkb:<http://bio2rdf.org/pharmgkb_vocabulary>
PREFIX omim:<http://bio2rdf.org/omim_vocabulary>
R1: SELECT ?drug ?hgnc ?model WHERE {
    ?drug  dbank:target ?trgt .
    ?trgt  dbank:x-hgnc ?hgnc .
    ?hgnc  hgnc:x-mgi ?marker .
    ?model mgi:marker ?marker .
    ?model mgi:allele ?all .
    ?all   mgi:allele-attribute ?allele_type .
    FILTER(?drug = <http://bio2rdf.org/drugbank:
                        DB00619>)
}

R2: SELECT ?gene ?ref ?cterm  where{
    <http://bio2rdf.org/pharmgkb:PA446359>
           pgkb:x-snomedct ?cterm .
    ?gene  rdf:type   omim:Gene .
    ?gene  omim:refers-to ?ref .
    ?ref   omim:x-snomed ?cterm .
}

R3: SELECT ?drug ?s ?protein WHERE
{
    ?drug  dbank:gene-name ?geneName.
    ?drug  dbank:x-uniprot ?protein.
    ?drug  dbank:general-function ?genFunction.
    ?drug  dbank:specific-function ?speFunction.
    ?pheno rdf:type omim:Phenotype .
    ?pheno rdfs:label ?o.
    ?pheno omim:clinical-features ?clinicFeature.
    ?pheno omim:article ?article.
    ?pheno omim:x-uniprot ?protein.
r} limit 500
```

V. The Drug Query Based on QFed

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/
          drugbank/resource/drugbank/>
PREFIX dosage: <http://www4.wiwiss.fu-berlin.de/
          drugbank/resource/dosageforms/>
PREFIX sider: <http://www4.wiwiss.fu-berlin.de/
          sider/resource/sider/>
PREFIX dailymed: <http://www4.wiwiss.fu-berlin.de/
          dailymed/resource/dailymed/>
SELECT * WHERE {
 ?disease rdfs:label "Asthma" .
 ?drug drugbank:possibleDiseaseTarget ?disease .
 ?drug  drugbank:dosageForm dosage:tabletOral .
 ?drug  drugbank:brandName ?brand_name .
 ?drug  rdfs:label ?drug_name .
 OPTIONAL {
  ?siderdrug owl:sameAs ?drug .
  ?siderdrug    sider:sideEffect ?sideeffect .
  ?sideeffect rdfs:label "Acidosis" .
  ?moiety rdfs:label ?drug_name .
  ?branded_drug dailymed:activeMoiety ?moiety .
  ?branded_drug  dailymed:contraindication ?contr .
  }
 }
```

VI. LUBM Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-
        syntax-ns\#>
PREFIX ub: <http://swat.cse.lehigh.edu/onto/
        univ-bench.owl\#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
        schema\#>
Q1: SELECT ?X ?Y ?Z WHERE{
    ?X rdf:type ub:GraduateStudent .
    ?Y rdf:type ub:University .
    ?Z rdf:type ub:Department .
    ?X ub:memberOf ?Z .
    ?Z ub:subOrganizationOf ?Y .
    ?X ub:undergraduateDegreeFrom ?Y . }

Q2: SELECT ?X ?Y ?Z WHERE{
    ?X rdf:type ub:GraduateStudent .
    ?Y rdf:type ub:AssociateProfessor .
    ?Z rdf:type ub:GraduateCourse .
    ?X ub:advisor ?Y .
    ?Y ub:teacherOf ?Z .
    ?X ub:takesCourse ?Z . }

Q3: SELECT ?X WHERE{
    ?X rdf:type ub:GraduateStudent .
    ?X ub:undergraduateDegreeFrom <www.University0.edu> .
    }

Q4: SELECT ?X ?Y ?Z ?N WHERE{
    ?X rdf:type ub:GraduateStudent .
    ?Y rdf:type ub:AssociateProfessor .
    ?Z rdf:type ub:GraduateCourse .
    ?X ub:advisor ?Y .
    ?Y ub:teacherOf ?Z .
    ?X ub:takesCourse ?Z .
    ?Y ub:undergraduateDegreeFrom ?U .
    ?U ub:name ?N .
}
```

VII. LargeRDFBench Queries

```
PREFIX drgbnk: <http://www4.wiwiss.fu-berlin.de/
        drugbank/resource/drugbank/>
PREFIX drgcat: <http://www4.wiwiss.fu-berlin.de/
        drugbank/resource/drugbank/drugcategory>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX bio: <http://bio2rdf.org/ns/bio2rdf#>
PREFIX biokegg: <http://bio2rdf.org/ns/kegg#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geonames: <http://www.geonames.org/ontology#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX tcga: <http://tcga.deri.ie/schema/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX nyt : <http://data.nytimes.com/elements/>
PREFIX chebi: <http://bio2rdf.org/ns/chebi#>
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tcgsch: <http://tcga.deri.ie/schema/>


S3: SELECT ?president ?party ?page WHERE {
    ?president rdf:type dbpedia:President .
    ?president dbpedia:nationality res:United_States .
    ?president dbpedia:party ?party .
    ?x nyt:topicPage ?page .
    ?x owl:sameAs ?president .
}
S11: SELECT ?drugDesc ?cpd ?equation WHERE {
    ?drug drgbnk:drugCategory drgcat:cathartics .
    ?drug drgbnk:keggCompoundId ?cpd .
    ?drug drgbnk:description ?drugDesc .
    ?enzyme biokegg:xSubstrate ?cpd .
    ?enzyme rdf:type biokegg:Enzyme .
    ?reaction biokegg:xEnzyme ?enzyme .
    ?reaction biokegg:equation ?equation .
}
S13: SELECT ?drug ?title WHERE {
   ?drug drgbnk:drugCategory drgcat:micronutrient.
   ?drug drgbnk:casRegistryNumber ?id .
   ?keggDrug rdf:type biokegg:Drug .
   ?keggDrug bio:xRef ?id .
```

```
   ?keggDrug dc:title ?title .
}


C2: SELECT ?drug ?keggmass ?chebiIupacName
WHERE
{
    ?drug rdf:type drgbnk:drugs .
    ?drug drgbnk:keggCompoundId ?keggDrug .
    ?keggDrug bio:mass ?keggmass .
    ?drug drgbnk:genericName ?drugBankName .
    ?chebiDrug purl:title ?drugBankName .
    ?chebiDrug chebi:iupacName ?chebiIupacName .
    OPTIONAL {
        ?drug drgbnk:inchiIdentifier ?drugbankInchi .
        ?chebiDrug bio2RDF:inchi ?chebiInchi .
        FILTER (?drugbankInchi = ?chebiInchi)
    }
}

C3: SELECT DISTINCT ?artist ?name ?location
    ?anylocation WHERE {
    ?artist a mo:MusicArtist .
    ?artist foaf:name ?name .
    ?artist foaf:based_near ?location .
    ?location geonames:parentFeature ?locationName .
    ?locationName geonames:name ?anylocation .
    ?nytLocation owl:sameAs ?location.
    ?nytLocation nytimes:topicPage ?news
    OPTIONAL
    {
        ?locationName geonames:name
            'Islamic Republic of Afghanistan' .
    }
}

C4: SELECT DISTINCT ?countryName ?countryCode
    ?locationMap ?population ?long ?lat ?anthem
    ?fDate ?largestCity ?ethGroup ?motto WHERE {
      ?NYTplace geonames:name ?countryName;
      geonames:countryCode ?countryCode;
      geonames:population  ?population;
      geo:long   ?long;
      geo:lat     ?lat;
      owl:sameAs   ?geonameplace.
      OPTIONAL {
          ?geonameplace dbpedia:capital ?capital;
          dbpedia:anthem ?anthem;
          dbpedia:foundingDate ?fDate;
          dbpedia:largestCity ?largestCity;
          dbpedia:ethnicGroup ?ethGroup;
          dbpedia:motto ?motto.
      }
}
LIMIT 50

B2: SELECT DISTINCT ?patient ?tumorType ?exonValue
WHERE
{
    ?s tcga:bcr_patient_barcode ?patient .
    ?patient tcga:disease_acronym
                <http://tcga.deri.ie/lusc> .
    ?patient tcga:tumor_weight ?weight .
    ?patient tcga:tumor_type ?tumorType .
    ?patient tcga:result ?results .
    ?results tcga:RPKM  ?exonValue .
    FILTER(?weight <= 55)
}

B3: SELECT ?patient ?methylationValue
WHERE
{
    ?s tcga:bcr_patient_barcode ?patient.
    ?patient tcgsch:vital_status "Dead".
    ?patient tcga:bcr_drug_barcode ?drug.
    ?drug tcga:drug_name "Tarceva".
    ?patient tcgsch:age_at_initial_pathologic_
                diagnosis ?age.
    ?patient tcga:result ?results.
    ?results  tcga:beta_value ?methylationValue.
    FILTER(?age <= 51)
}
ORDER BY (?patient)

B7: SELECT DISTINCT ?patient ?p ?o WHERE
{
    ?uri tcga:bcr_patient_barcode ?patient .
    ?patient dbpedia:country ?country.
    ?country dbpedia:populationDensity ?popDensity.
    ?patient tcga:bcr_aliquot_barcode ?aliquot.
    ?aliquot ?p ?o.
    FILTER(?popDensity >= 32)
}
```

VIII. QFED QUERIES

```
PREFIX sider: <http://www4.wiwiss.fu-berlin.de/
          sider/resource/sider/>
PREFIX diseasome: <http://www4.wiwiss.fu-berlin.de/
          diseasome/resource/diseasome/>
prefix owl: <http://www.w3.org/2002/07/owl#>
C2P2BOF:
SELECT * {
    ?s1 a sider:side_effects .
    ?s1 owl:sameAs ?s2 .
    ?s2 diseasome:associatedGene ?URI .
    OPTIONAL {
      ?s2 diseasome:classDegree ?LITERAL .
    }
    FILTER(?LITERAL >= 3) .
}
```

REFERENCES

[1] D. DeWitt and J. Gray. Parallel database systems: The future of high performance database systems. *Communications of the ACM*, 35(6), 1992.

[2] R. Harbi, I. Abdelaziz, P. Kalnis, N. Mamoulis, Y. Ebrahim, and M. Sahli. Accelerating sparql queries by exploiting hash-based locality and adaptive partitioning. *The VLDB Journal*, 2016.