Architectures and algorithms for Web data in the cloud

Ioana Manolescu

OAK team, Inria Saclay and Université de Paris Sud

Joint work with A. Aranda Andújar, F. Bugiotti, J. Camácho-Rodríguez, D. Colazzo, F. Goasdoué and Z. Kaoudi

Plan

1. A few words about the OAK team
2. Warehousing Web data in the Amazon cloud
   - Architecture
   - Some experimental results
3. Other ongoing work
   1. Efficient RDF data management on Hadoop
   2. Translating complex query languages to PACT (with TU Berlin)
   3. Storing and indexing a mesh of views in the cloud

OAK team in short

- Joint team Inria Saclay – LRI, Université de Paris Sud
- Permanent members
  - Ioana Manolescu (DR Inria)
  - Nicole Bidoit (PR Paris Sud)
  - Dario Colazzo (MdC Paris Sud, HDR)
  - François Goasdoué (MdC Paris Sud, HDR)
  - Melanie Herschel (MdC Paris Sud)
- Plus 8 PhD students, 3 young Inria engineer, 1 post-doc

- Research on database optimizations and architectures for complex large data

OAK topics overview

- Optimizations for large-scale XML data management
- Scalable complex processing for RDF data
  - Growing
- Models and optimizations for emerging data models
  - Annotated documents, temporal models
- Architecture and optimization for complex data in the cloud
- Architectures for data transformations on large, evolving datasets
  - Advanced ETL: query analysis, maintenance, repair
- All related to Labex DigiCosme Task 1 (Scalable, secure data management)

Storing Web data in the Amazon cloud

Web data

- Increasingly important for organizations
  - Recognize the need for scalable platforms to manage it
- Large volumes of data are produced using XML or RDF

Linked Open Data

BBC

data.gov.uk BETA

DBpedia

DATA.GOV
EMPOWERING PEOPLE

RDF

Cloud computing

- Scalability
- Elasticity
- Fault-tolerance
- No hardware administration

Provided services?

At what cost?

Goal

- **Investigate architectures for scalable Web data stores based on off-the-shelf commercial cloud services**
  - Scale up to large data volumes
  - Efficient document storage and querying operations
  - Minimize resource usage ➡ In the cloud, total work translates into monetary costs

Proposed architecture: Amada

- Use of an standard XML query processor to extract the results from the documents
- Indexes help to decide which documents are concerned by a given query
  - No need to run the query processor over the complete data set
    - Reduces **total work** for answering a query
    - This *may* reduce **response time**
    - It *always* reduces **monetary costs** of running the data store
  - Multiple indexing strategies with different levels of detail

  - ACM CIKM 2012: http://team.saclay.inria.fr/oak/amada
  - Data Analytics in the Cloud workshop (DanaC) 2012, with EDBT
  - Data Management in the Cloud workshop (DMC) 2012, with ICDE

Amazon Web Services (AWS)

SimpleDB

- Storing and querying small, structured data

Simple Storage Service (S3)

- Storage for raw data, ideal for large files

Elastic Compute Cloud (EC2)

- Resizable computing capacity in the cloud

Simple Queue Service (SQS)

- Queues for communication between distributed components

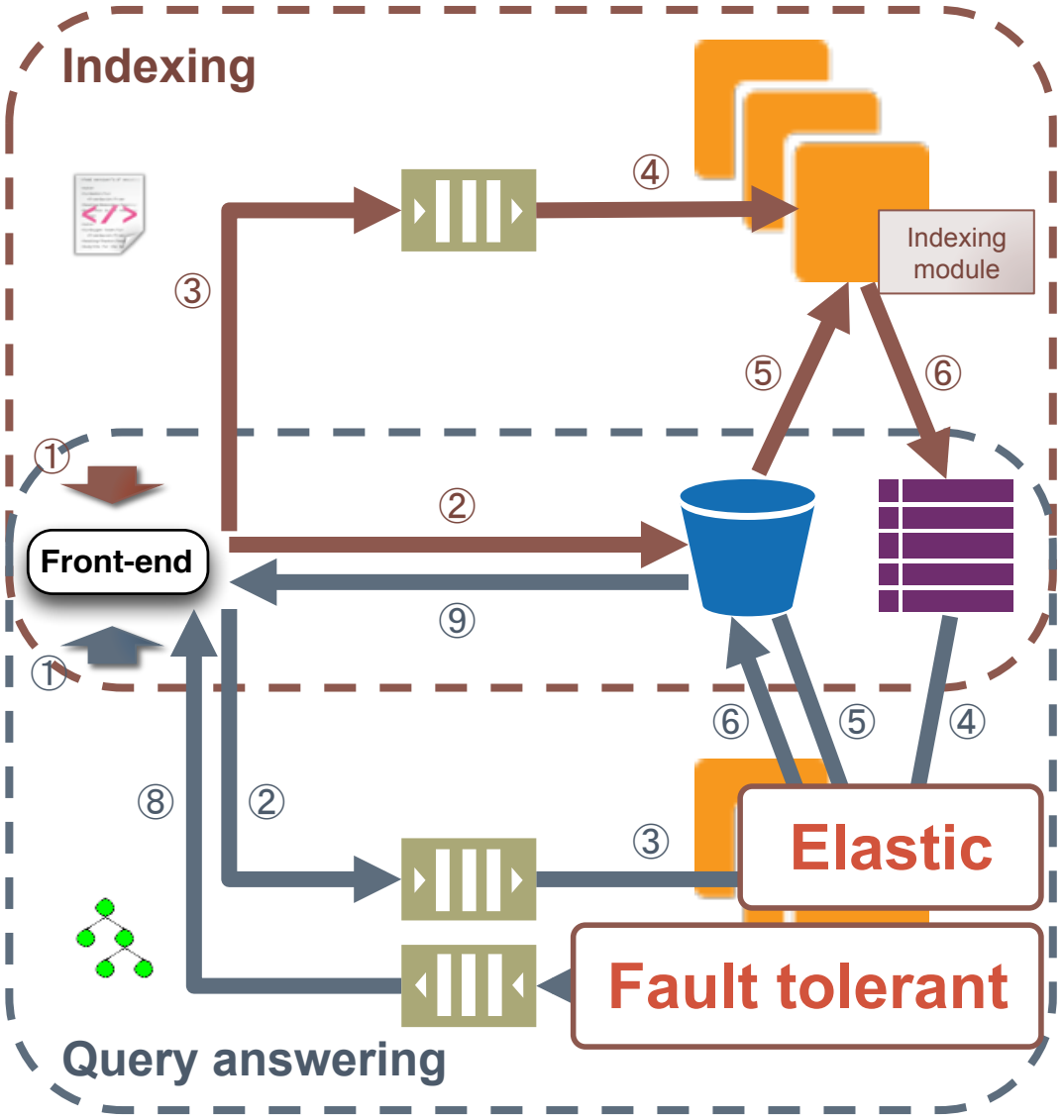
Amada architecture

DynamoDB
- Storing and querying small, structured data

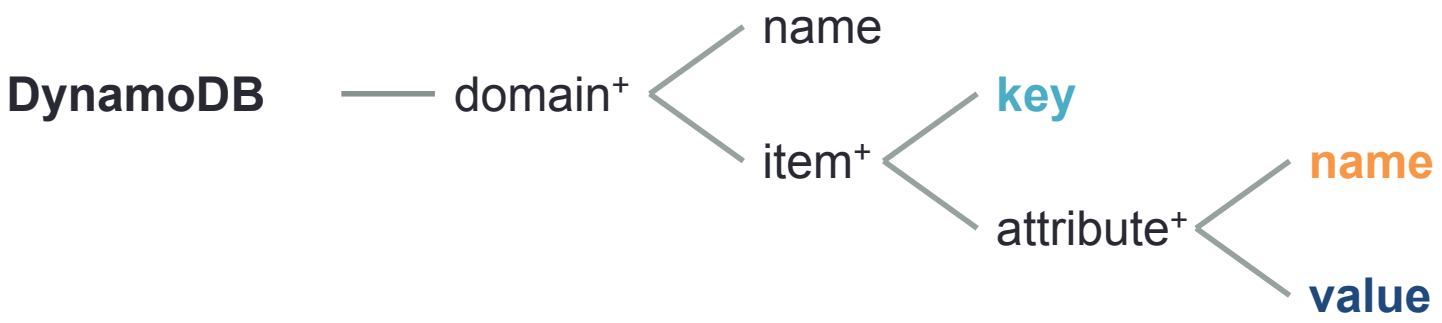
Simple Storage Service (S3)
- Storage for raw data, ideal for large files

Elastic Compute Cloud (EC2)
- Resizable computing capacity in the cloud

Simple Queue Service (SQS)
- Queues for communication between distributed components



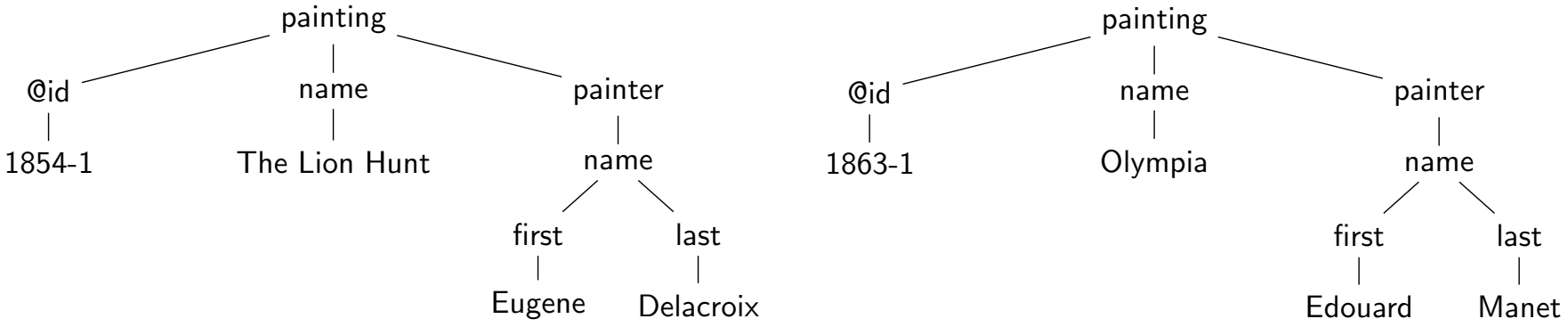
Indexing strategies



Indexing strategy *I*: Function associating (**key**,(**name**,**value**)+)+ to a document

- Four strategies with different trade-offs
  - LU strategy (Label-URI)
  - LUP strategy (Label-URI-Path)
  - LUI strategy (Label-URI-ID)
  - 2LUPI strategy (Label-URI-Path, Label-URI-ID)

LU strategy (Label-URI)



ename	delacroix.xml	manet.xml
	∅	∅

aid	delacroix.xml	manet.xml
	∅	∅

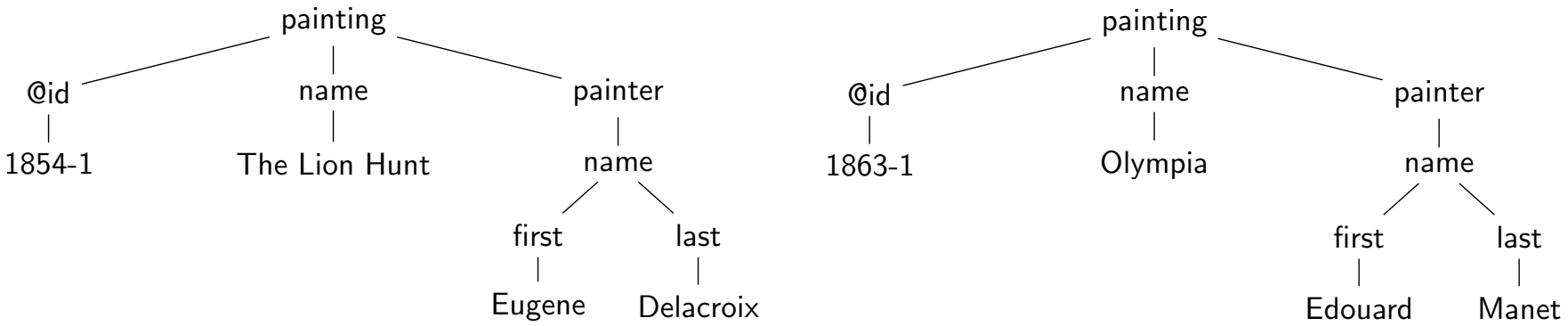
aid 1863-1	manet.xml
	∅

wOlympia	manet.xml
	∅

...

- Look-up: Intersection of URI sets associated to each query node

LUP strategy (Label-URI-Path)



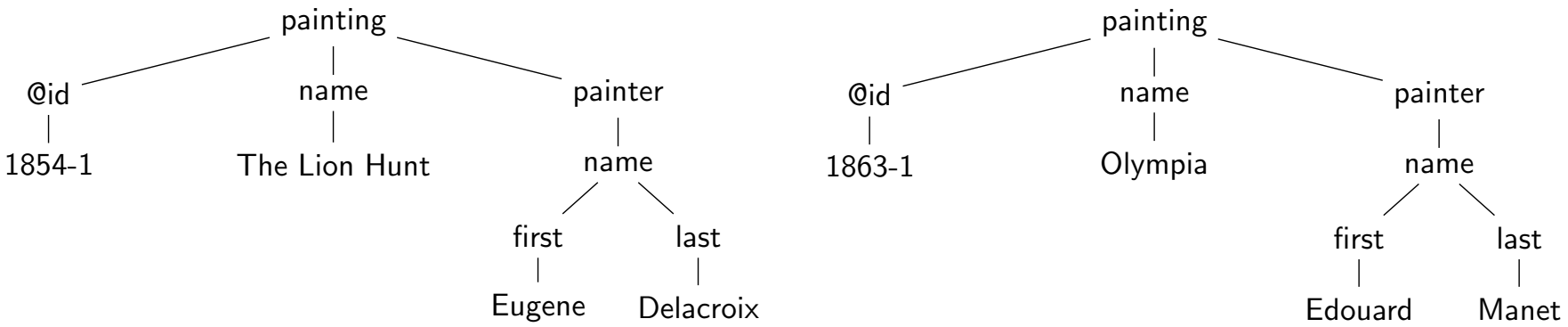
ename	delacroix.xml	delacroix.xml	manet.xml	manet.xml
	/epainting	/epainting/epainter	/epainting	/epainting/epainter

aid	delacroix.xml	manet.xml
	/epainting	/epainting

...

- Look-up: Match each query path with the paths associated to its last node

LUI strategy (Label-URI-ID)



ename	delacroix.xml	manet.xml
	[24 50 1][61 120 2]	[24 45 1][56 112 2]

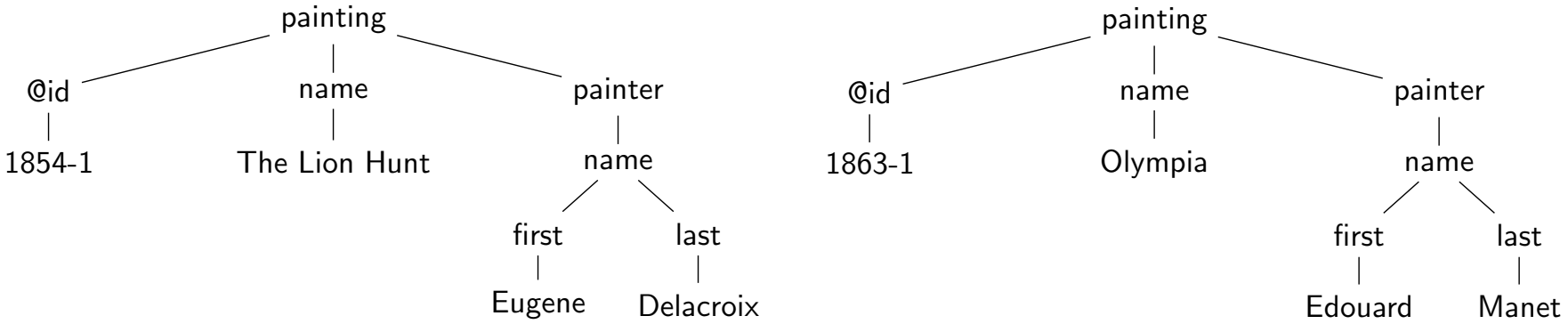
aid 1863-1	manet.xml
	[14 20 3]

aid	delacroix.xml	manet.xml
	[10 21 2]	[10 21 2]

...

- Look-up: Structural join over IDs associated to each query node

2LUPI strategy (Label-URI-Path, Label-URI-ID)



ename	delacroix.xml	delacroix.xml	manet.xml	manet.xml
	/epainting	/epainting/epainter	/epainting	/epainting/epainter

ename	delacroix.xml	manet.xml
	[24 50 1][61 120 2]	[24 45 1][56 112 2]

...

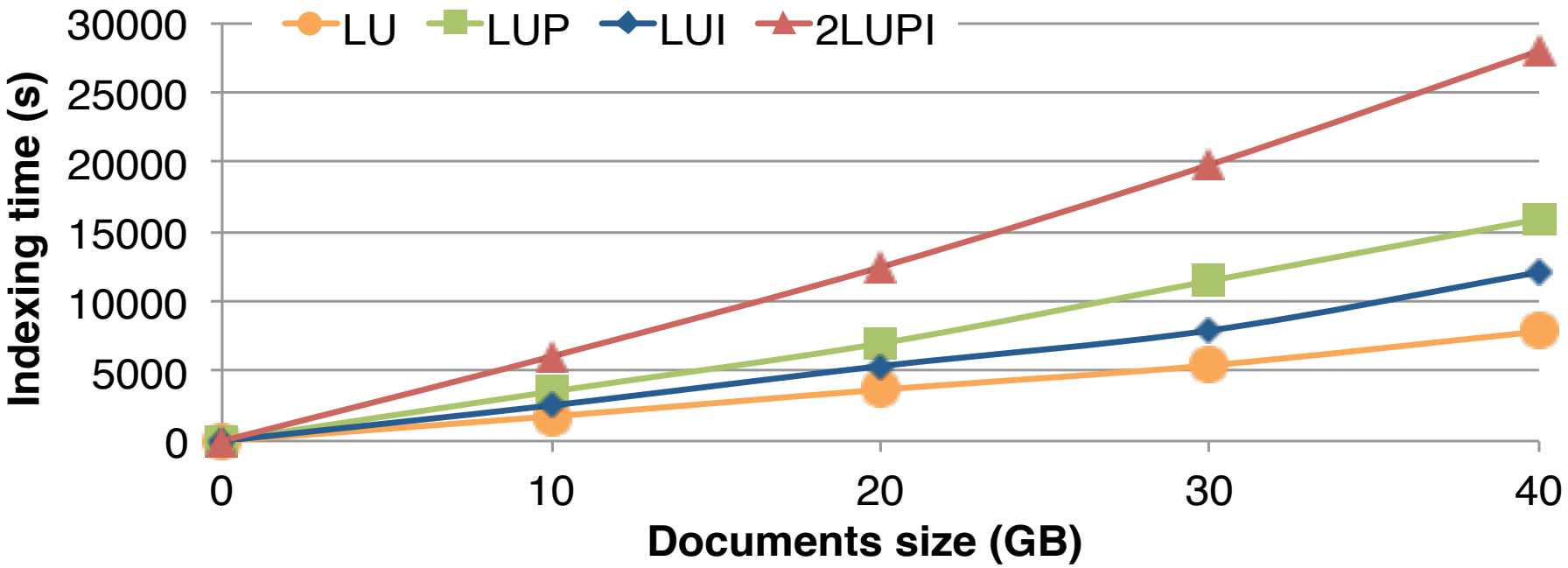
- Look-up: Obtain URIs using LUP and prefilter IDs using them, then structural join over the remaining IDs

Experiments in the Amazon cloud

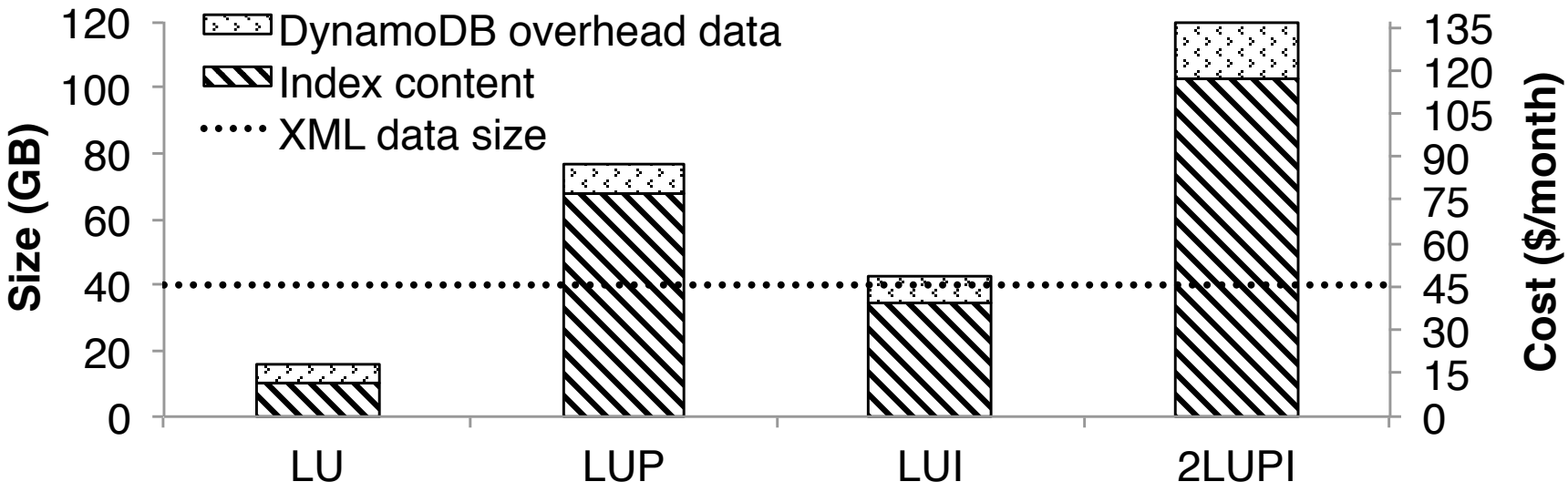
- System fully implemented in Java 6
  - Amazon Web Services SDK for Java v1.2.14
  - XML query processor from our ViP2P project[1]
- Two types of EC2 instances
  - Large (L), 7.5 GB of RAM memory and 2 virtual cores
  - Extra large (XL), 15 GB of RAM memory and 4 virtual cores
- XMark documents (≈40GB)
  - ⅓ original
  - ⅓ altered path structure
  - ⅓ more optional paths
- Workload (*WL*) consisted of 10 XMark (XML benchmark) queries

[1] http://vip2p.saclay.inria.fr/

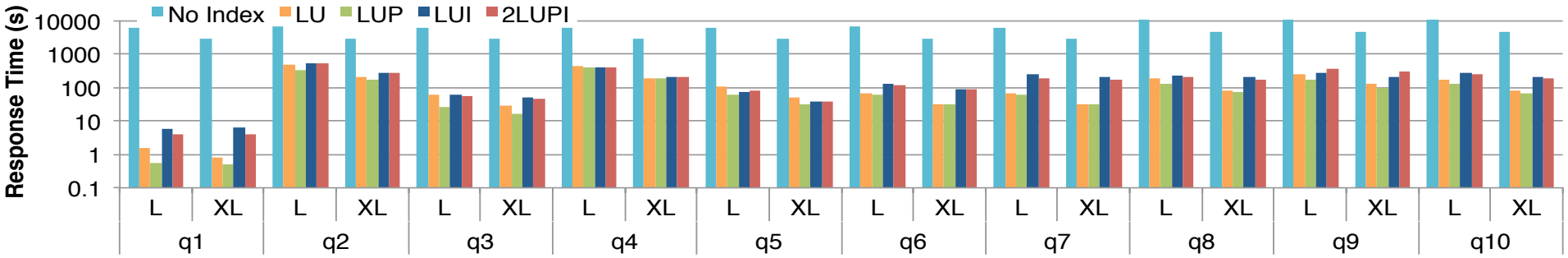
Index creation time (40 GB XML)



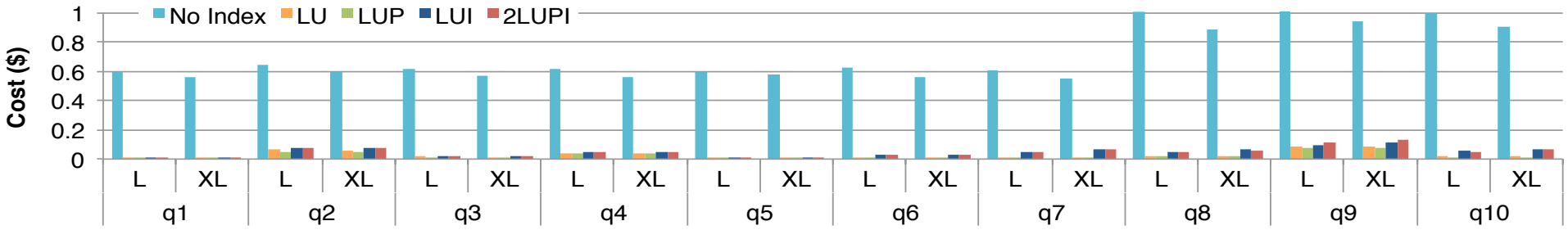
Index size and cost (40 GB XML)



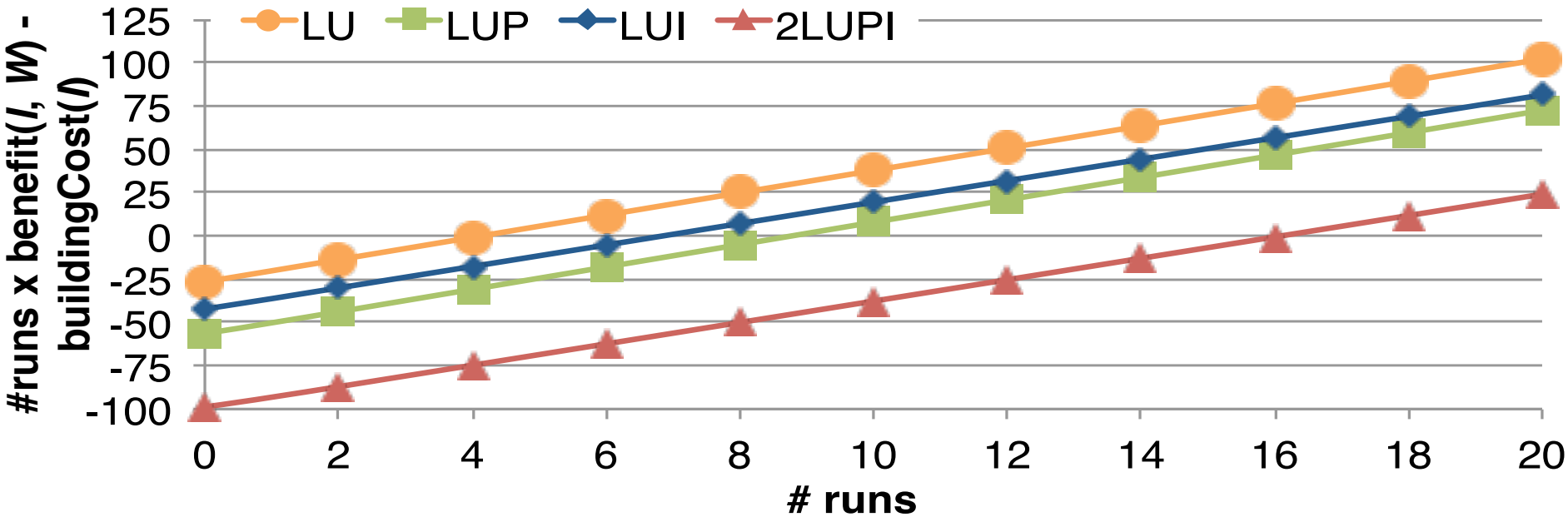
Query answering time (40 GB XML)



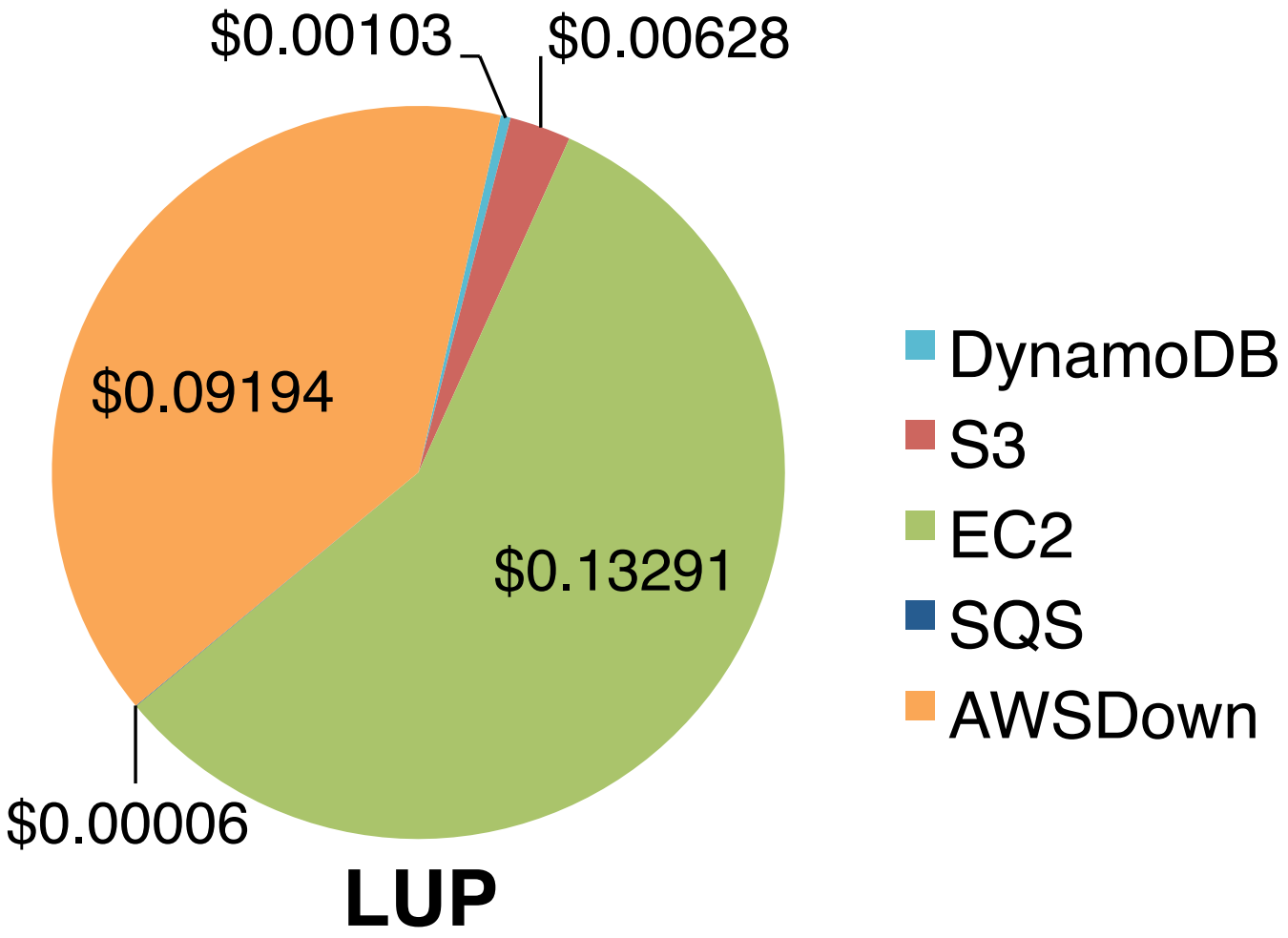
Query processing costs



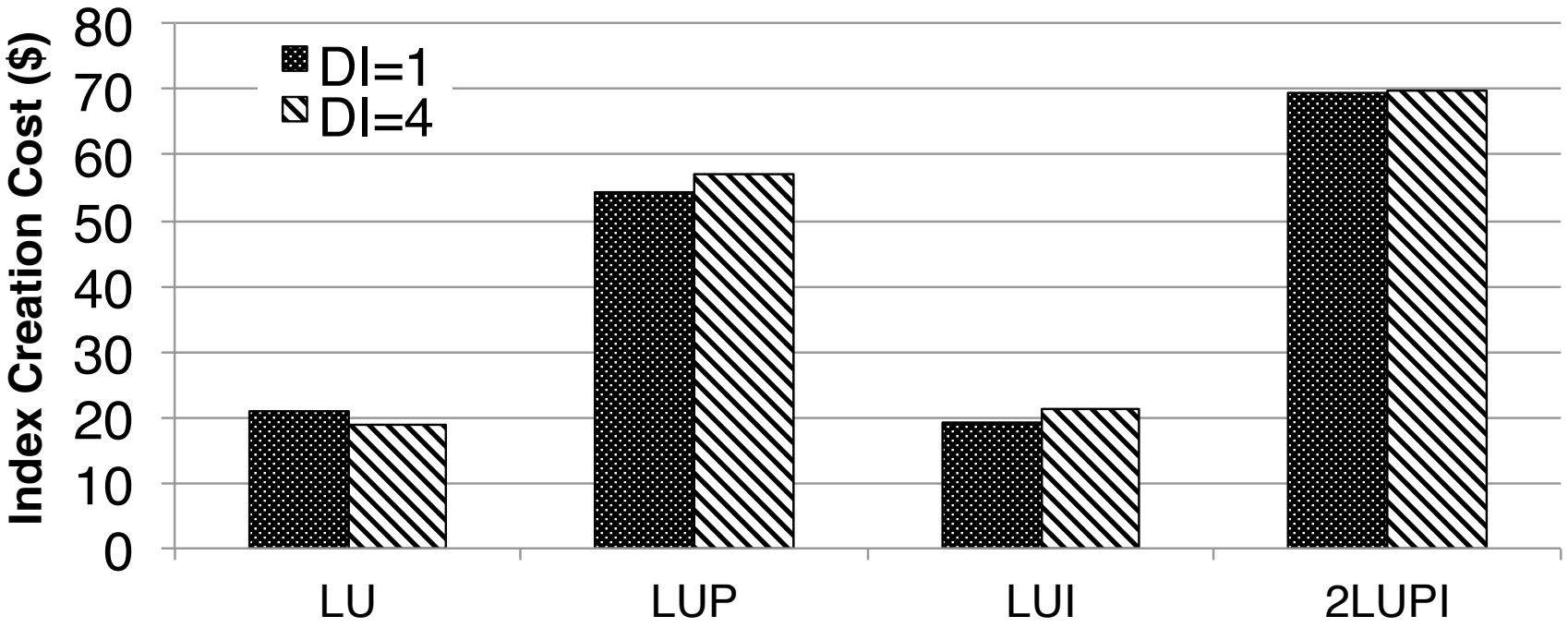
Index cost amortization



Cost distribution across operations



Impact of parallelism



Wrap-up: Web data stores in AWS

- AWS-based architecture
  - DynamoDB to store XML indexes
  - S3 to store XML documents
  - EC2 to create indexes and process queries
- Several indexing strategies
- Algorithm to distribute the index over multiple domains
- Experimental evaluation
  - Indexing strategies reduce query response time and monetary cost
- Related project on **RDF** data
  - More difficult: RDF database = single graph (after "merging")

Related works

- Large scale XML intra-query parallelism [FLG+11, KCS11]
- Commercial relational databases ported to the cloud: Oracle Database, IBM DB2 or Microsoft SQL Server
- Monetary costs in the cloud
  - OLTP on top of multiple commercial providers [KKL10]
  - Data-based services from providers perspective [KDF+11, KDG+11]

Other cloud-related works in the OAK team

Efficient RDF stores on Hadoop

- Problem: RDF is a single large big graph
- To scale, use Map/Reduce
  - But: this requires partitioning
  - Several existing approaches
  - We investigate a new one based on some redundancy of the store and on custom optimizations brought to Hadoop
- In collaboration with Jorge Quiané-Ruiz from the Qatar Computing Research Institute (PhD Inria)

- Tutorial at ICDE 2013 with Zoi Kaoudi "Triples in the Cloud"

Running complex queries on PACT

- PACT = generalization of Map/Reduce from the Technical University of Berlin
  - Library of massively parallel operators that extend Map and Reduce
  - Most notably with several inputs
  - PACT = PArallelization contraCT
    - Known properties of the operators' input
    - Constraints which hold on the operators' outputs
- PACT very suited for tabular data
- We work within the KIC EIT ICT Labs Europa activity, on translating complex queries (XML) into PACT programs

THANK YOU

More info: https://team.inria.fr/oak/cloud-data-management/

Questions?

[FLG+11] L. Fegaras, C. Li, U. Gupta, and J. Philip.
XML Query Optimization in Map-Reduce.
In WebDB, 2011.

[KCS11] S. Khatchadourian, M. P. Consens, and J. Siméon.
Having a ChuQL at XML on the cloud.
In A. Mendelzon Int'l. Workshop, 2011.

[KDF+11] V. Kantere, D. Dash, G. François, S. Kyriakopoulou, and A. Ailamaki.
Optimal service pricing for a cloud cache.
In IEEE Trans. Knowl. Data Eng., vol. 23, no. 9, 2011.

[KDG+11] V. Kantere, D. Dash, G. Gratsias, and A. Ailamaki.
Predicting cost amortization for query services.
In SIGMOD, 2011.

[KKL10] D. Kossmann, T. Kraska, and S. Loesing.
An evaluation of alternative architectures for transaction processing in the cloud.
In SIGMOD, 2010.