PERFORMANCE OF RDF QUERY PROCESSING ON THE INTEL SCC

Vasil Slavov, Praveen Rao, Dinesh Barenkala, Srivenu Paturi

Department of Computer Science & Electrical Engineering

University of Missouri-Kansas City, USA

Acknowledgements

Single-chip Cloud Computer Research Program, Intel Labs

National Science Foundation (IIS-1115871)

Overview

- Introduction
- Background
  - RDF and SPARQL
  - Prior work on RDF query processing
- Our methodology
- Performance evaluation
- Conclusion and future work

UMKC

Introduction (1/2)

- The RDF data model
  - Important in domain-specific applications and the WWW
    - Knowledge representation and reasoning
    - E.g., healthcare, defense and intelligence, biopharmaceuticals, Linked Data
- Very large RDF datasets are available today
  - DBPedia [WWW ‘07, ISWC ‘07]
  - YAGO2 [WWW ‘11]
  - Billion Triples Challenge (http://challenge.semanticweb.org)
  - Uniprot RDF (http://www.uniprot.org/downloads)

Introduction (2/2)

- Pressing need for high performance RDF processing tools

Can a manycore processor boost the performance of RDF query processing through parallel processing?

Background (1/3)

- RDF
  - Triple format: (subject, predicate, object)
  - Represents a directed, labeled graph

<u>Example</u>

<Intel_Corporation> rdf:type <wikicategory_Companies_established_in_1968> .
<Intel_Corporation> rdf:type <wikicategory_Motherboard_companies> .
<Intel_Corporation> rdf:type <wikicategory_Multinational_companies> .
<Intel_Corporation> rdf:type <wikicategory_Netbook_manufacturers> .
<Intel_Corporation> rdf:type <wikicategory_Semiconductor_companies> .
<Intel_Corporation> y:created <IA-32_Execution_Layer> .
<Intel_Corporation> y:created <Itanium> .
<Intel_Corporation> y:created <Light_Peak> .

Background (2/3)

- SPARQL
  - A popular query language for RDF

Example

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix y: <http://www.mpii.de/yago/resource/> .

SELECT ?a ?n WHERE
{
  ?a rdf:type <wikicategory_Motherboard_companies>  .
  ?a y:created ?n .
}
```

?a	?n
<Intel_Corporation>	<IA-32_Execution_Layer>
<Intel_Corporation>	<Itanium>
<Intel_Corporation>	<Light_Peak>

UMKC

Background (3/3)

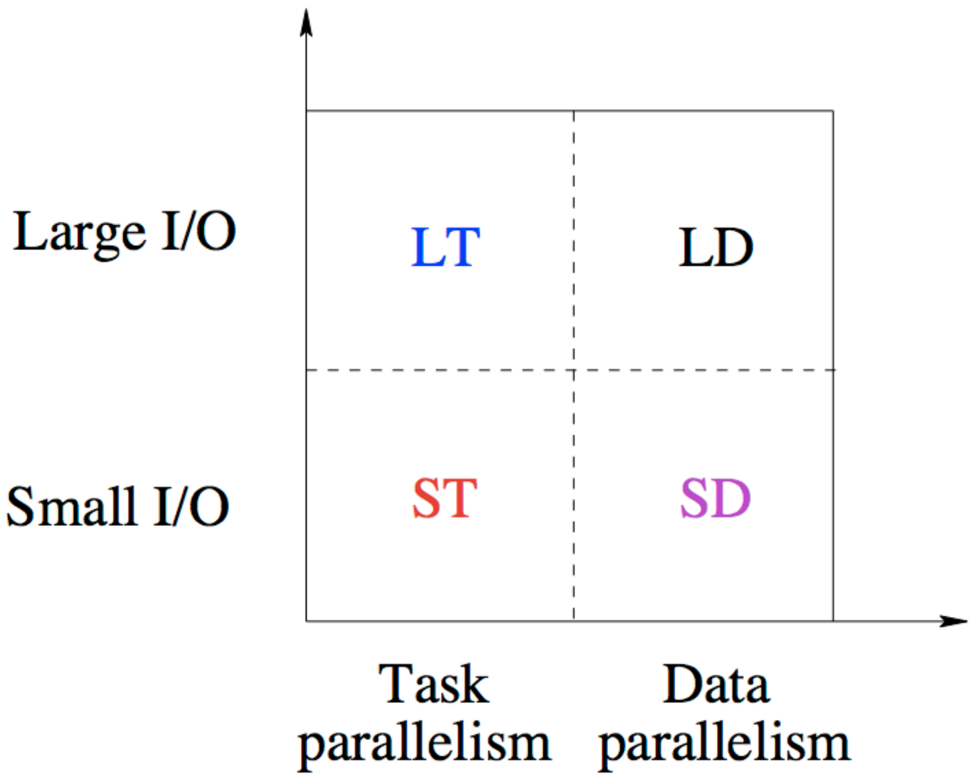
- Prior work on RDF query processing
  - Using an RDBMS
    - Sesame [ISWC ‘02]
    - Jena2 [SWDB ’03]
    - RDF_MATCH [VLDB ‘05]
    - Vertical partitioning [VLDB ‘07]
  - Native RDF databases
    - Hexastore [VLDB ‘08]
    - RDF-3X [VLDB ‘08, VLDB Journal ‘10]
    - BitMat [WWW ‘10]
  - Shared-nothing clusters
    - YARS2 [ISWC ‘07], 4store
    - Tools built using Apache Hadoop and Apache Pig
- On the Intel SCC (but not RDF)
  - Parallel AI planning [MARC ‘11]
  - Relational decision support queries [MARC ‘11]

UMKC

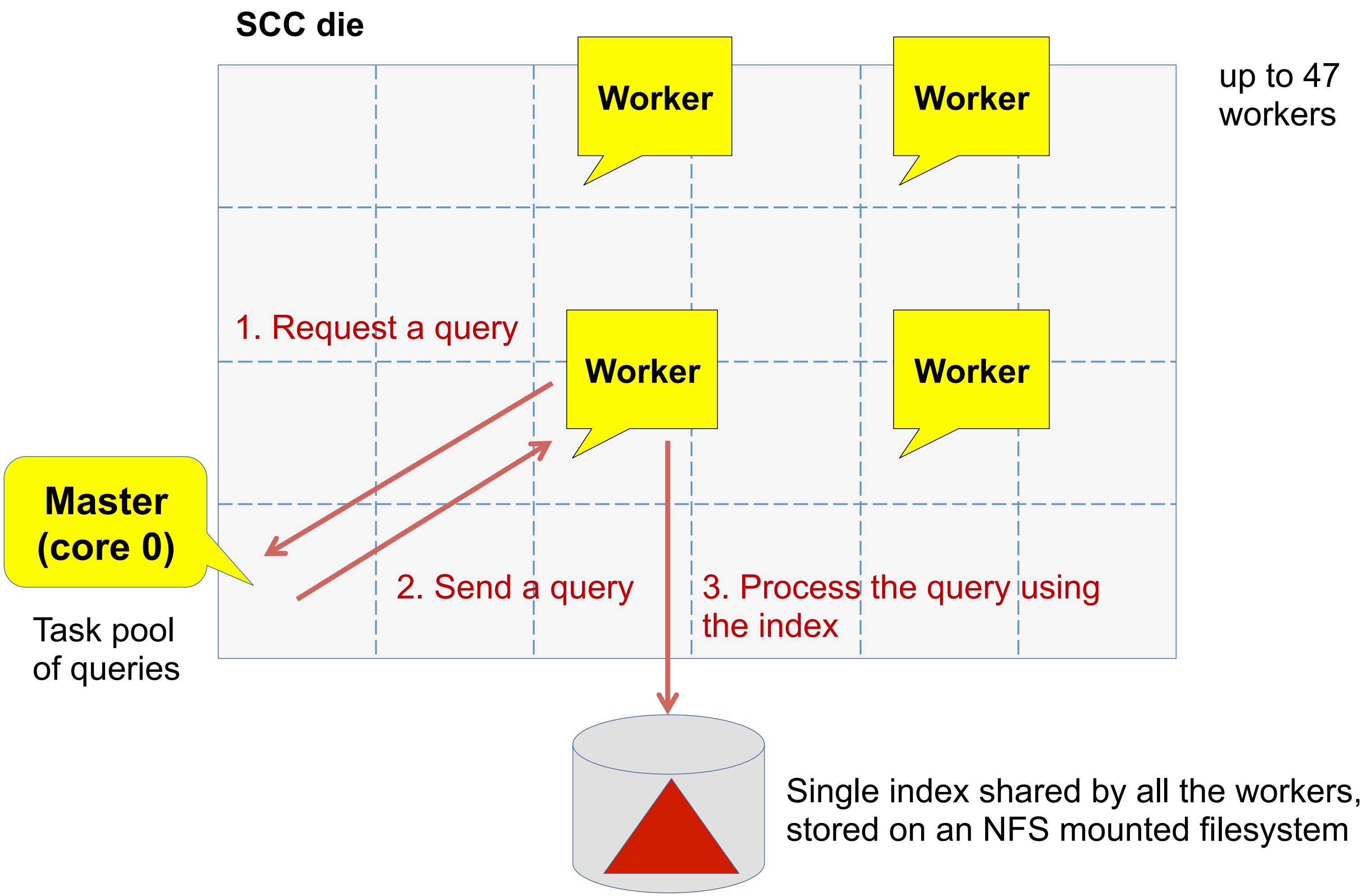
Our Methodology

- Programming models
  - Task (inter-query) parallelism
  - Data (intra-query) parallelism
- I/O bound queries
  - Small I/O footprint
  - Large I/O footprint
- Message Passing Interface (MPI)
  - MPI_Send, MPI_Recv
  - MPI_Barrier
  - MPI_Bcast, MPI_Scatter, MPI_Gatherv

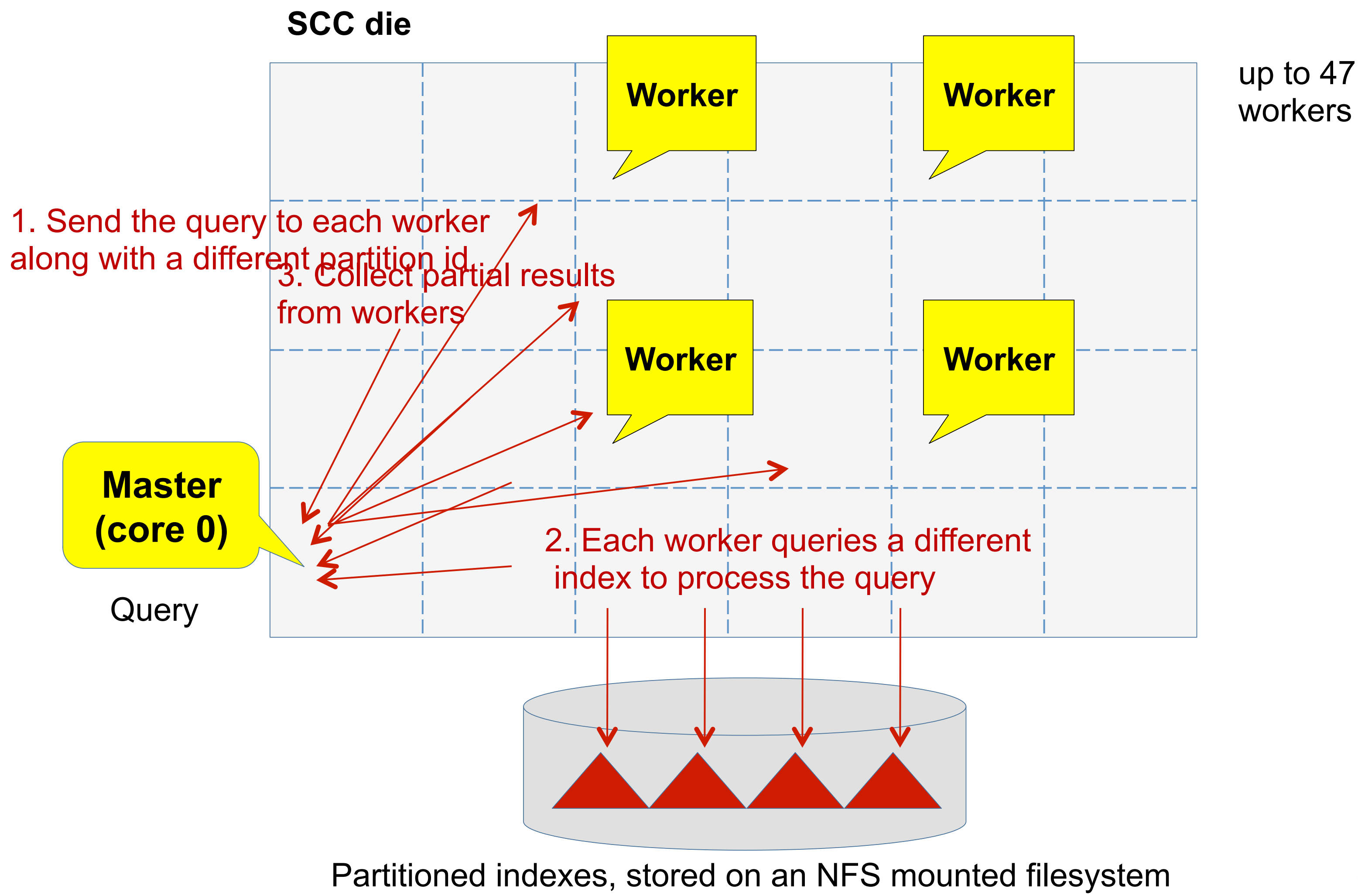
Models


Large I/O
Small I/O
LT
LD
ST
SD
Task parallelism
Data parallelism

Task Parallel Programming Model



Data Parallel Programming Model (1/2)



Data Parallel Programming Model (2/2)

- Partition the RDF graph into smaller graphs
  - Extract weakly connected directed subgraphs
  - Apply standard graph partitioning techniques (e.g., METIS [SIAM '98])
  - We may miss results
- Collect partial results
  - Multiple MPI_Recv
  - Single MPI_Gatherv

Performance Evaluation

- RDF-3X [VLDB ‘08, VLDB Journal ‘10]
- RCKMPI (a modified MPICH2 for Intel SCC)
- Tile_Mesh_DDR: 800MHz, 800MHz, 800MHz
- 2GB index size (limit of OS)
- Indexes – stored on an NFS mounted filesystem
- Cold cache

Datasets

- Real datasets
  - YAGO2 (27,331,797 triples)
  - Uniprot (46,972,851 triples)
- Synthetic dataset
  - LUBM (35,612,176 triples) [WWW ‘05]
- Data partitioning
  - LUBM (based on RDF files)
  - Uniprot (based on protein fragments)
  - YAGO2
    - Star-shaped graphs
    - METIS

YAGO: Queries

Query	I/O footprint	Type	% CPU	Serial time
QY_1	14,756 KB	small	29	4.73 secs
QY_2	15,004 KB	small	40	9.23 secs
QY_3	22,832 KB	small	29	6.51 secs
QY_4	33,492 KB	small	21	9.27 secs
QY_5	216,564 KB	large	22	82.65 secs
QY_6	272,848 KB	large	30	120.08 secs
QY_7	332,944 KB	large	43	218.43 secs

UMKC

LUBM: Queries

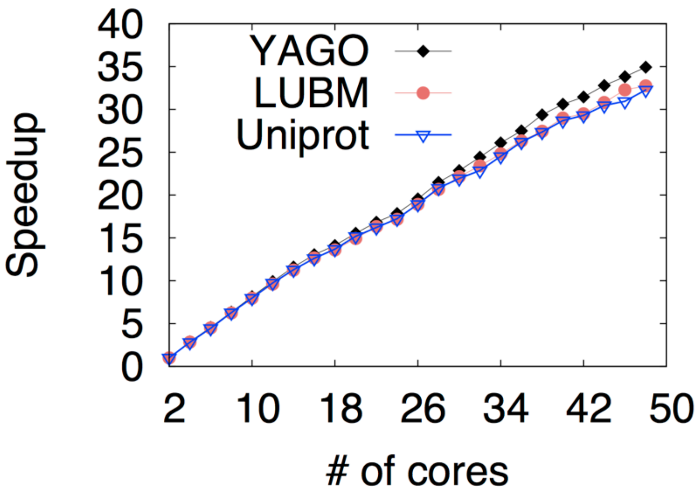
Query	I/O footprint	Type	% CPU	Serial time
QL_1	2,668 KB	small	25	1.4 secs
QL_2	3,132 KB	small	35	1.47 secs
QL_3	9,804 KB	small	19	3.5 secs
QL_4	636,204 KB	large	32	299.99 secs
QL_5	673,924 KB	large	29	206.58 secs

Uniprot: Queries

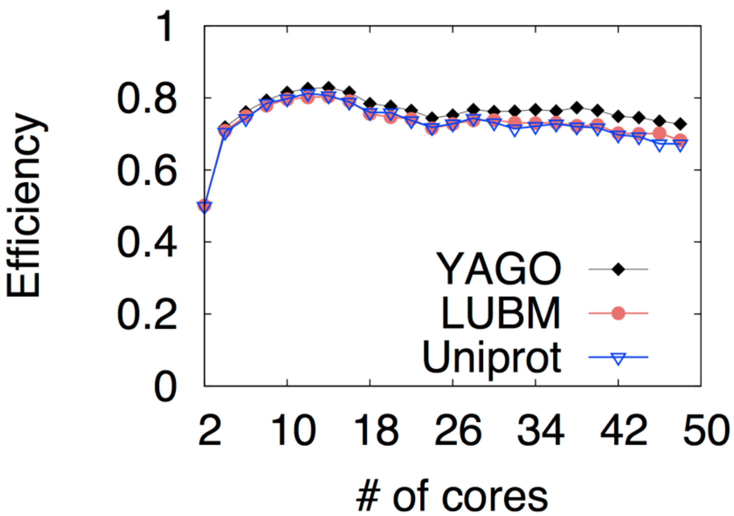
Query	I/O footprint	Type	% CPU	Serial time
QU_1	4,468 KB	small	39	2.08 secs
QU_2	10,344 KB	small	39	6.46 secs
QU_3	48,020 KB	large	31	19.39 secs
QU_4	62,188 KB	large	19	15.48 secs
QU_5	166,808 KB	large	17	43.51 secs

ST Model (small I/O footprint, task parallelism)

Speedup

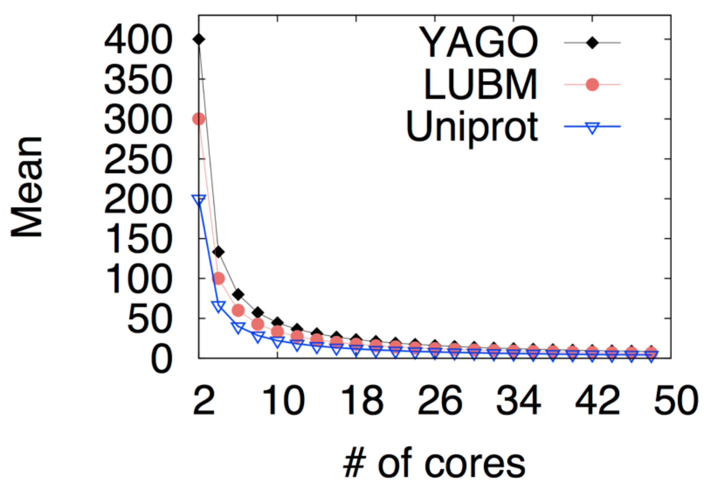


Efficiency

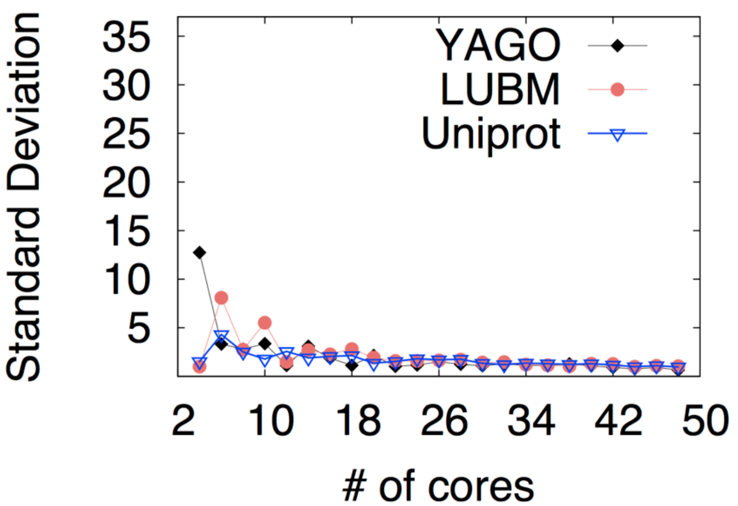


ST Model Load Distribution

Mean (# of tasks/worker)

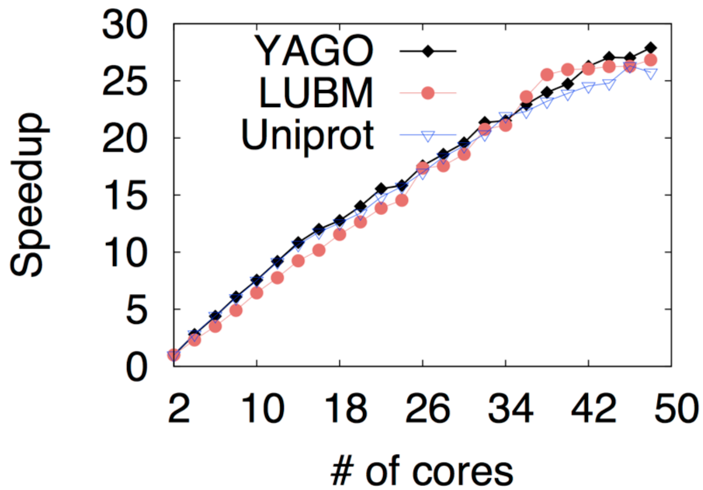


Standard deviation

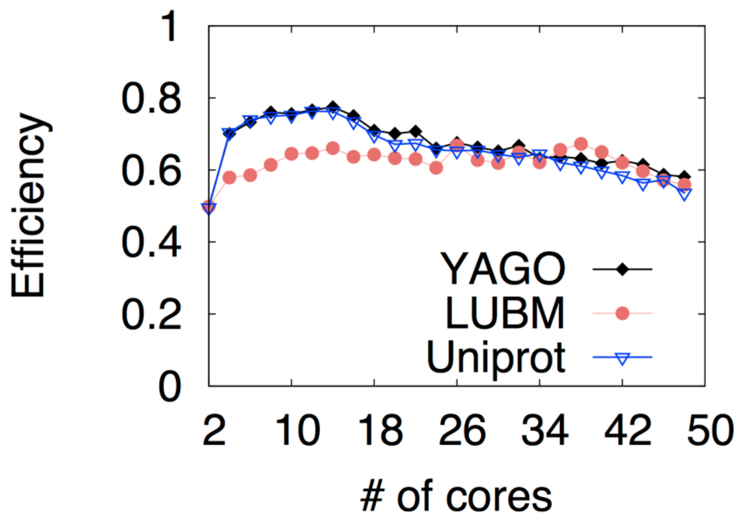


LT Model (large I/O footprint, task parallelism)

Speedup

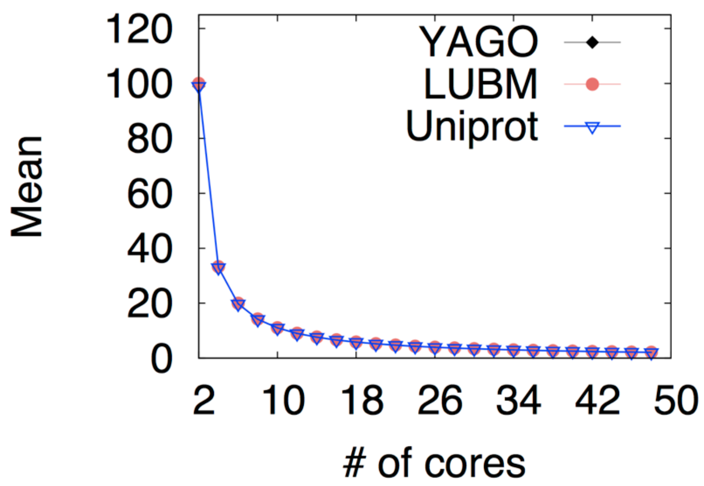


Efficiency

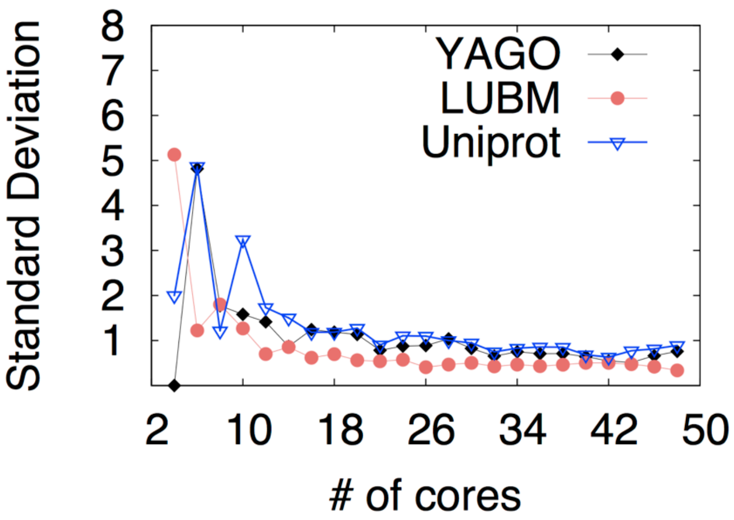


LT Model Load Distribution

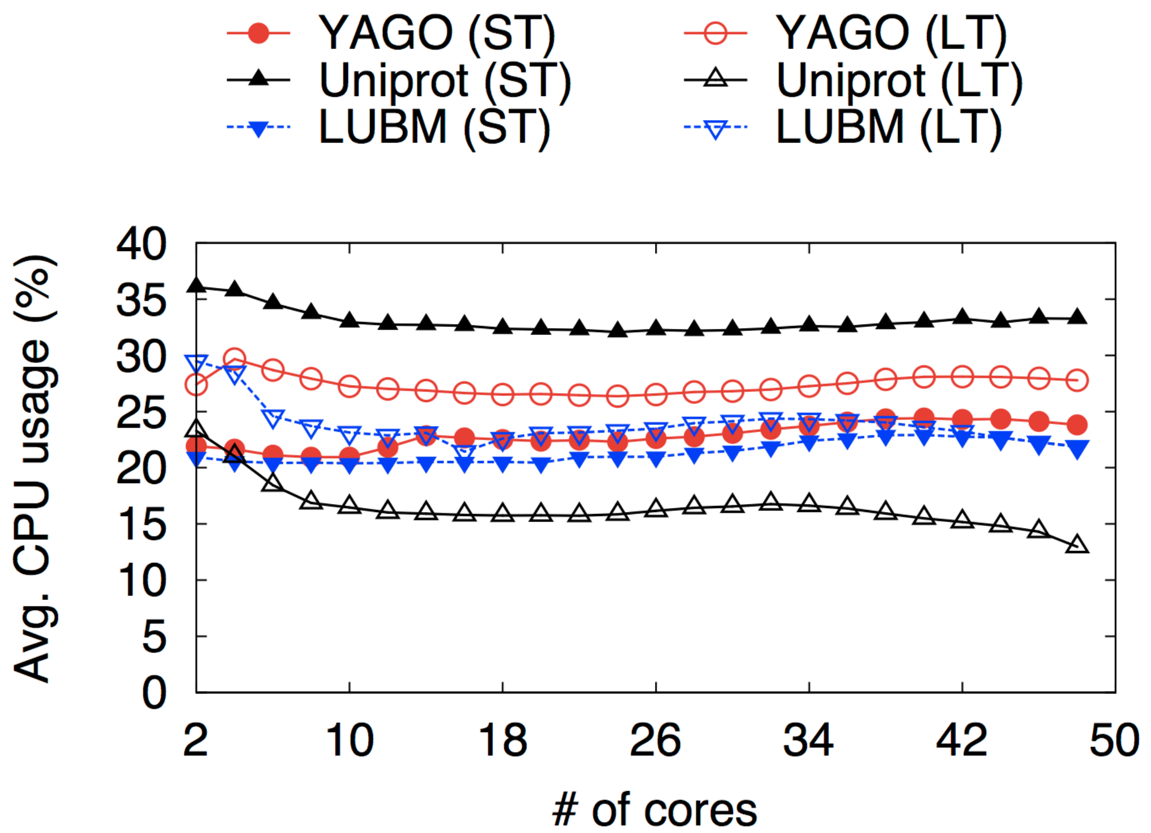
Mean (# of tasks/worker)



Standard deviation

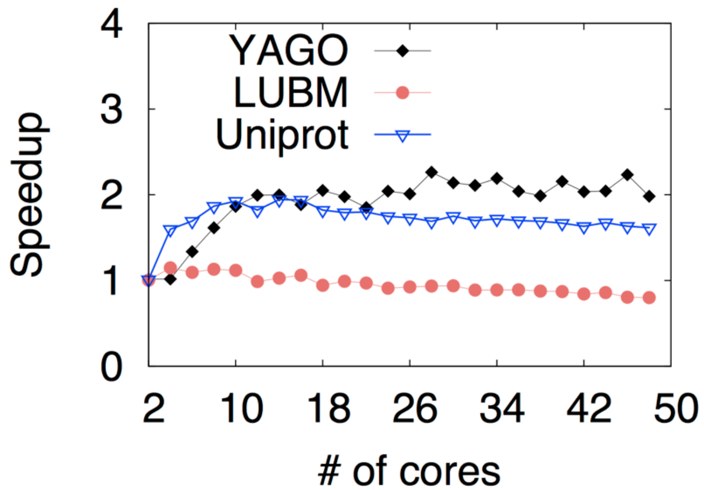


CPU Usage

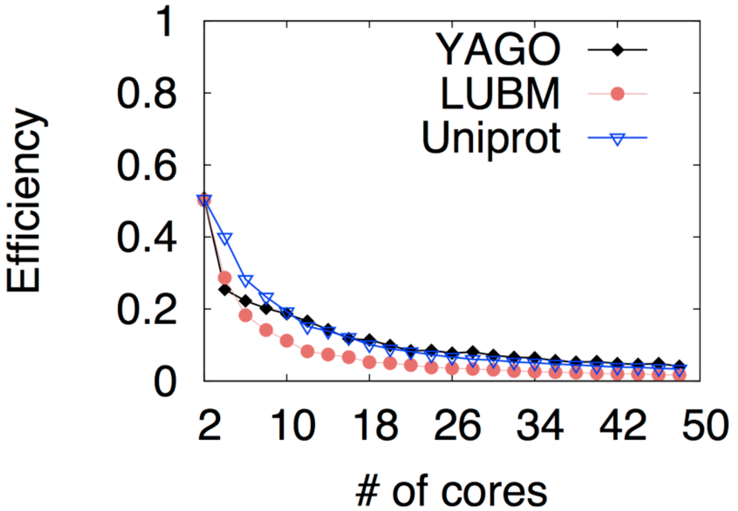

YAGO (ST)
YAGO (LT)
Uniprot (ST)
Uniprot (LT)
LUBM (ST)
LUBM (LT)
Avg. CPU usage (%)
40
35
30
25
20
15
10
5
0
2
10
18
26
34
42
50
# of cores

SD Model (small I/O footprint, data parallelism)

Speedup

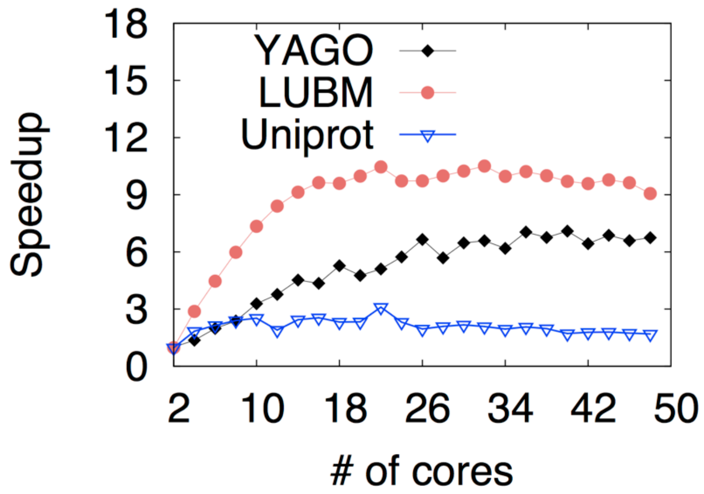


Efficiency

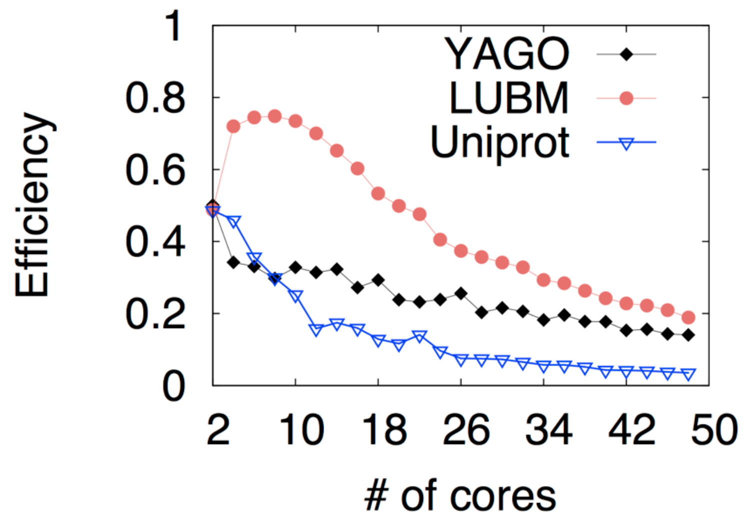


LD Model (large I/O footprint, data parallelism)

Speedup



Efficiency



Conclusion and Future Work

- Task parallel programming yields good speedup and efficiency
  - For both large I/O and small I/O footprint queries
- Data parallel programming yields poor speedup and efficiency due to
  - Load imbalance or I/O contention
- Future work
  - New methods to overcome the challenges posed by the data parallel programming model
  - Effect of dynamic voltage and frequency scaling on the performance of RDF query processing

UMKC

Questions?

- Acknowledgements
  - Single-chip Cloud Computer Research Program, Intel Labs
  - National Science Foundation (IIS-1115871), 2011-2014
- For further information, contact Prof. Praveen Rao (raopr@umkc.edu)