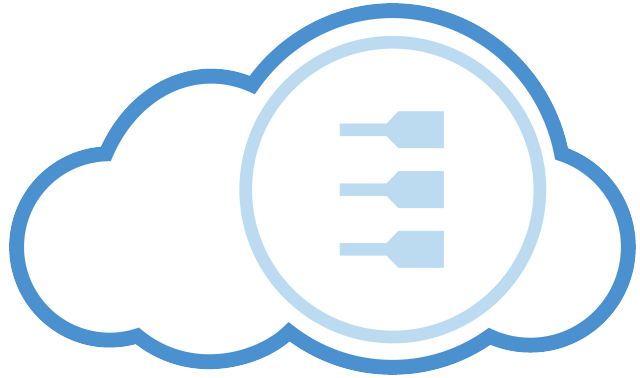


When to Reach for the Cloud

Using Parallel Hardware for Link Discovery

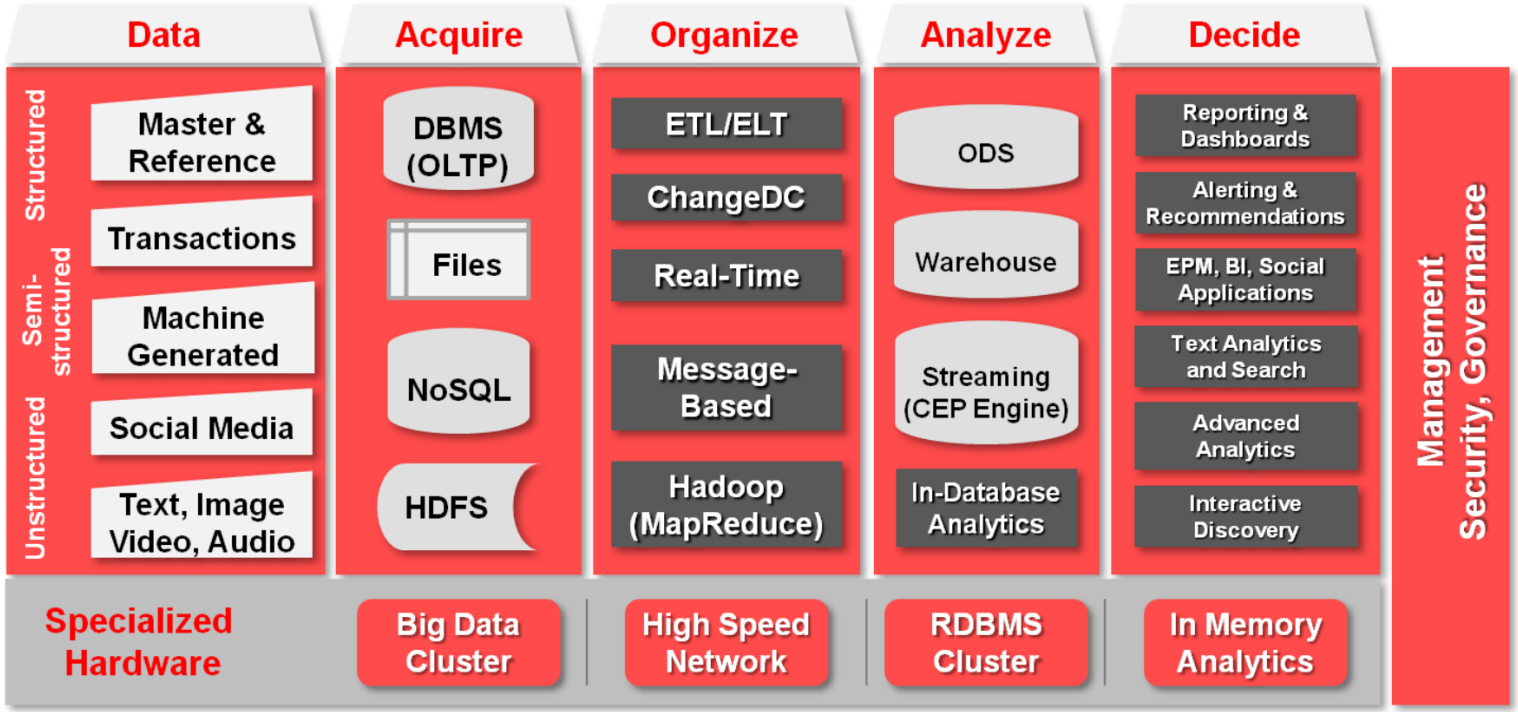
Axel-Cyrille Ngonga Ngomo Lars Kolb Norman Heino
Michael Hartung Sören Auer Erhard Rahm

ISO 15926 and Semantic Technologies, Sogndal, Norway

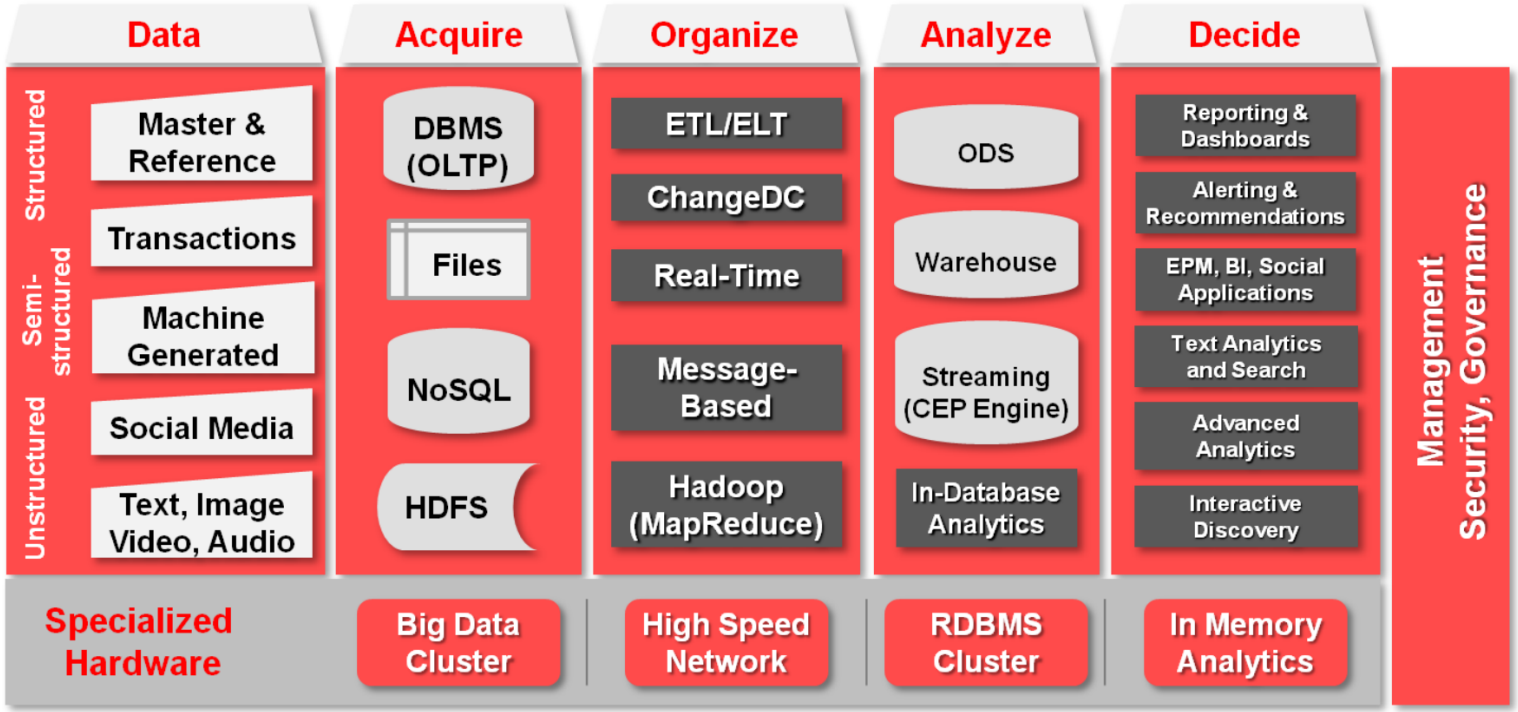
Outline

Outline

Big Data Processing



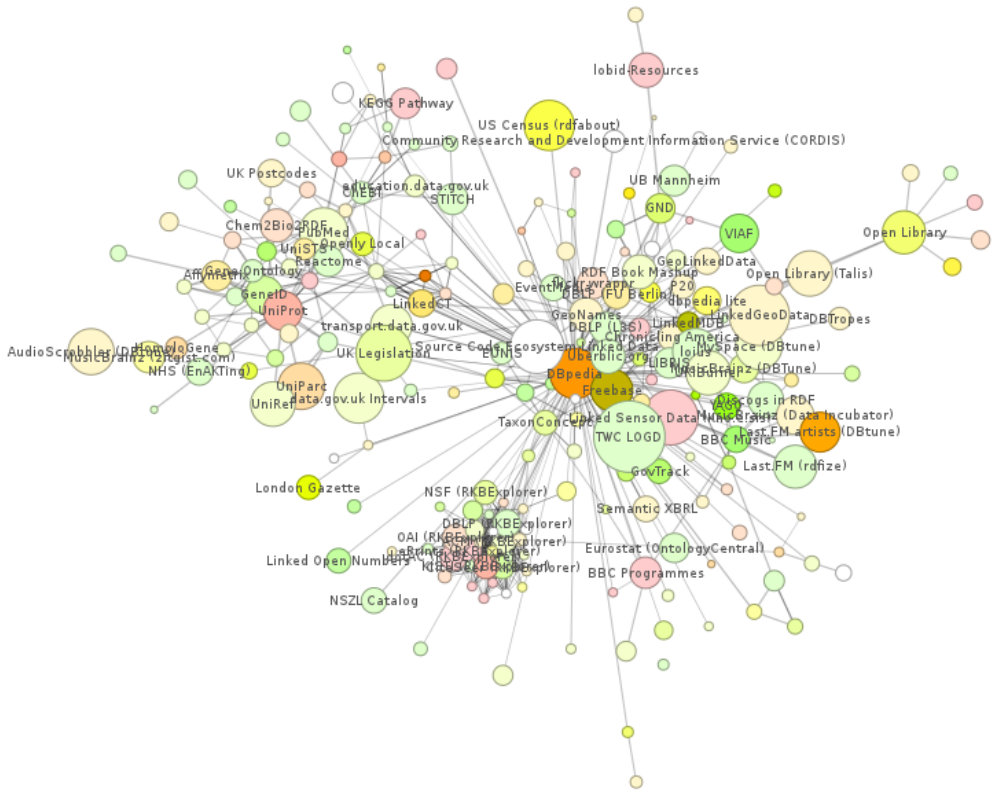
Big Data Processing



- Tendency towards porting all processing steps to the Cloud
- Full results available at `http://big-project.eu`
- **Question**: When should we reach to the cloud?

Why Link Discovery?

1. Fourth principle
2. Links are central for
   - **Data Integration**
   - Cross-ontology QA
   - Reasoning
   - Federated Queries
   - ...
3. Current topology of the LOD Cloud
   - 31+ billion triples
   - $\approx$ 0.5 billion links
   - `owl:sameAs` in most cases



Why is it difficult?

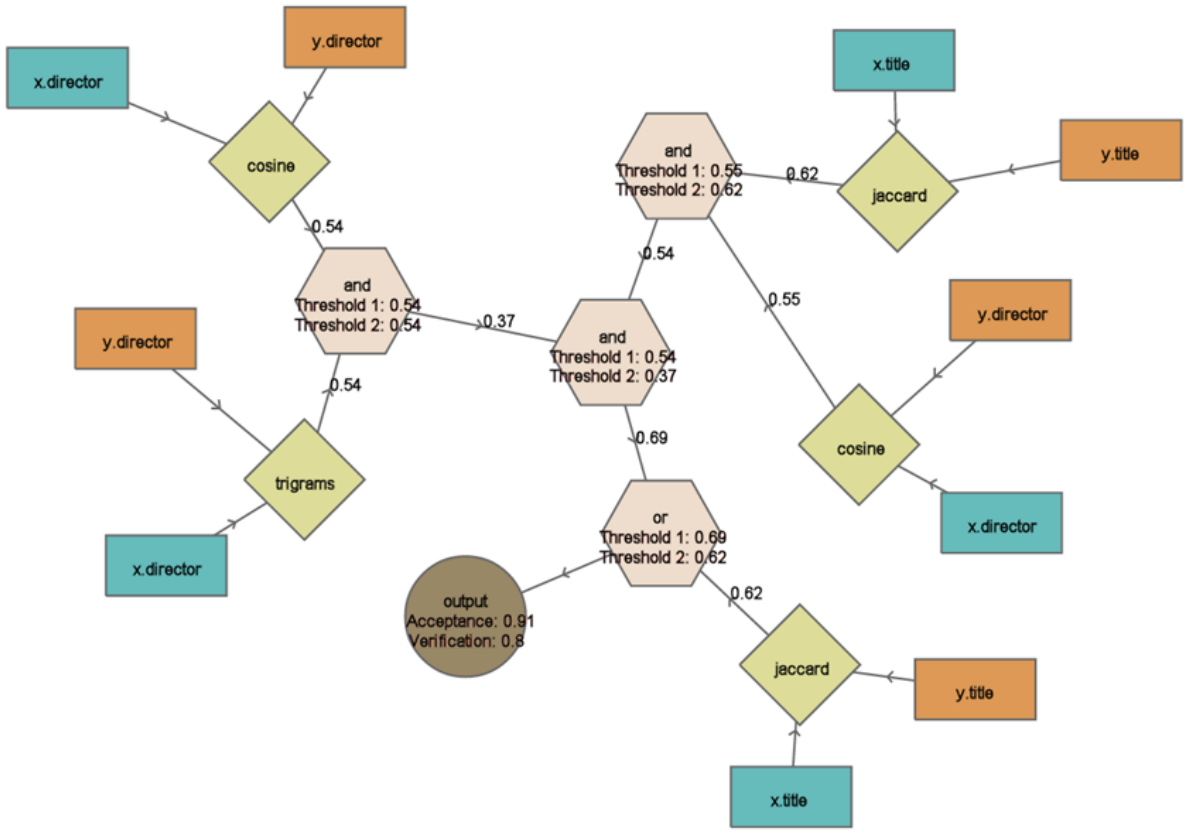
1. **Time complexity**
   - Large number of triples
   - Quadratic a-priori runtime
   - 69 days for mapping cities from DBpedia to Geonames

Definition (Link Discovery)

- Given sets S and T of resources and relation $\mathcal{R}$
- Find $M = \{(s,t) \in S \times T : \mathcal{R}(s,t)\}$
- Common approaches:
  - Find $M' = \{(s,t) \in S \times T : \sigma(s,t) \geq \theta\}$
  - Find $M' = \{(s,t) \in S \times T : \delta(s,t) \leq \theta\}$

Why is it difficult?

2. **Complexity of specifications**
   - Combination of several attributes required for high precision
   - Tedious discovery of most adequate mapping
   - Dataset-dependent similarity functions



Dealing with Time Complexity

1. **Devise better algorithms**

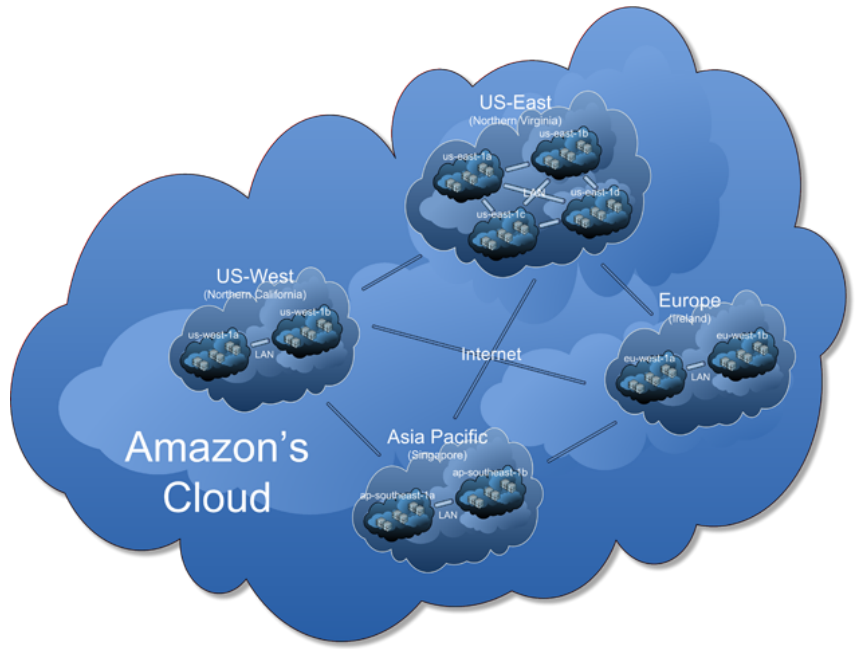
- Blocking
- Algorithms for given metrics
  - PPJoin+, EDJoin
  - $\mathcal{HR}^3$

Dealing with Time Complexity

1 Devise better algorithms

- Blocking
- Algorithms for given metrics
  - PPJoin+, EDJoin
  - $\mathcal{HR}^3$

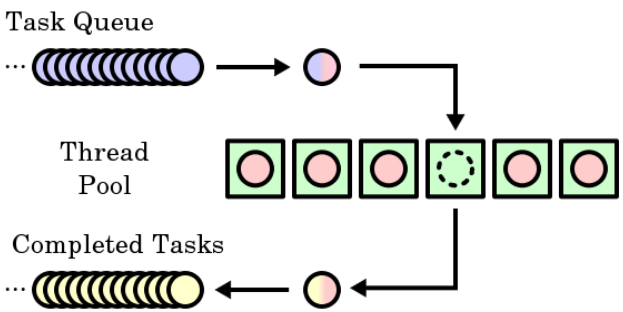
2 Use parallel hardware

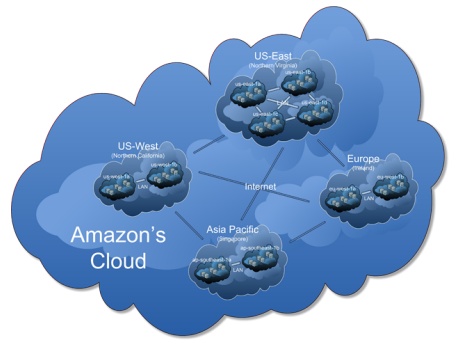


- Threads pools
- MapReduce
- Massively Parallel Hardware

Parallel Implementations

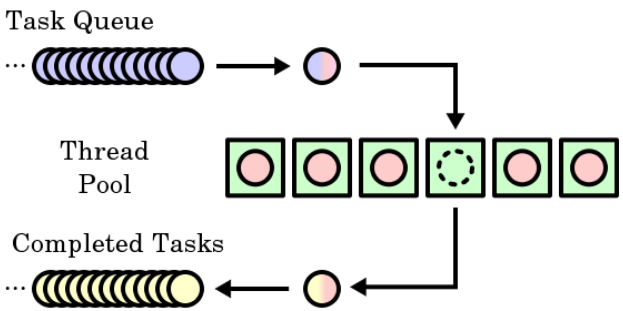
- Different architectures
  - Memory (shared, hybrid, distributed)
  - Execution paths (different, same)
  - Location (remote, local)

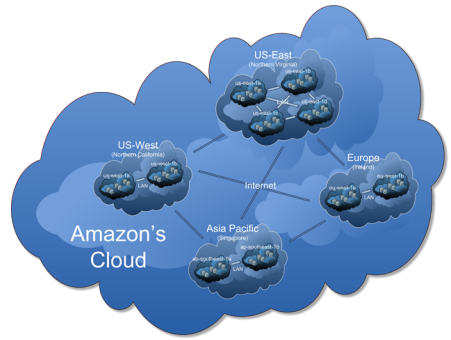




Parallel Implementations

- Different architectures
  - Memory (shared, hybrid, distributed)
  - Execution paths (different, same)
  - Location (remote, local)





Question

- When should which use which hardware?

Outline

Premises

- Given an algorithm that runs on all three architectures ...

Premises

- Given an algorithm that runs on all three architectures ...
- **Note to self**: Implement one
  - Picked $\mathcal{HR}^3$
  - Reduction-ratio-optimal

Premises

- Given an algorithm that runs on all three architectures ...
- **Note to self**: Implement one
  - Picked $\mathcal{HR}^3$
  - Reduction-ratio-optimal

Goals

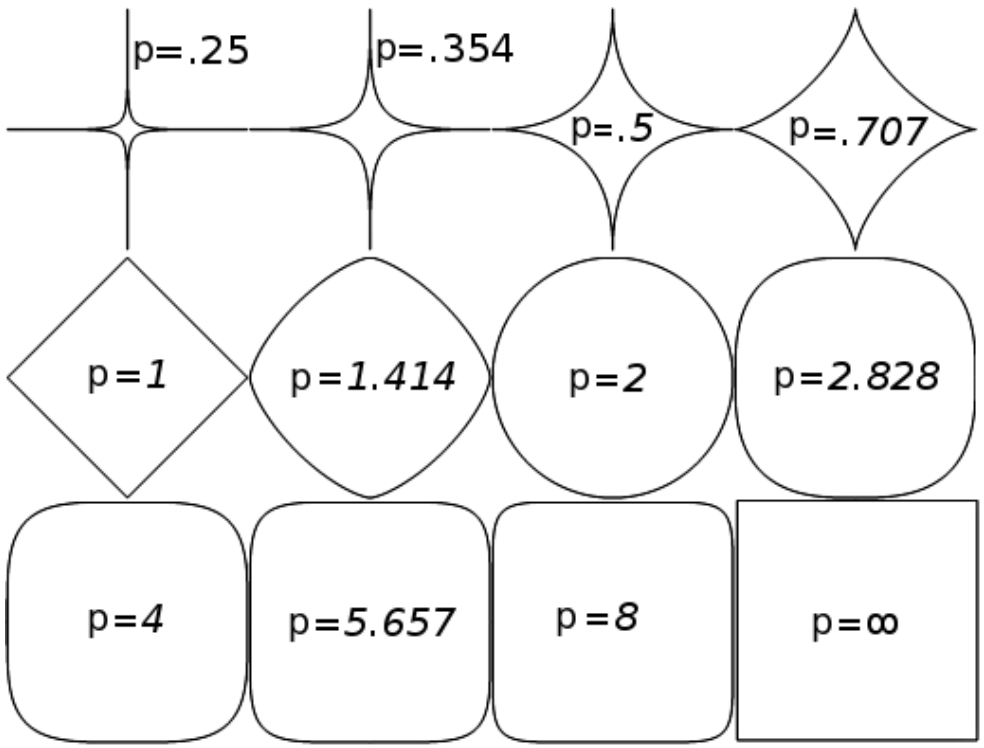
- Compare runtimes on all three parallel architectures
- Find break-even points

Outline

$\mathcal{HR}^3$

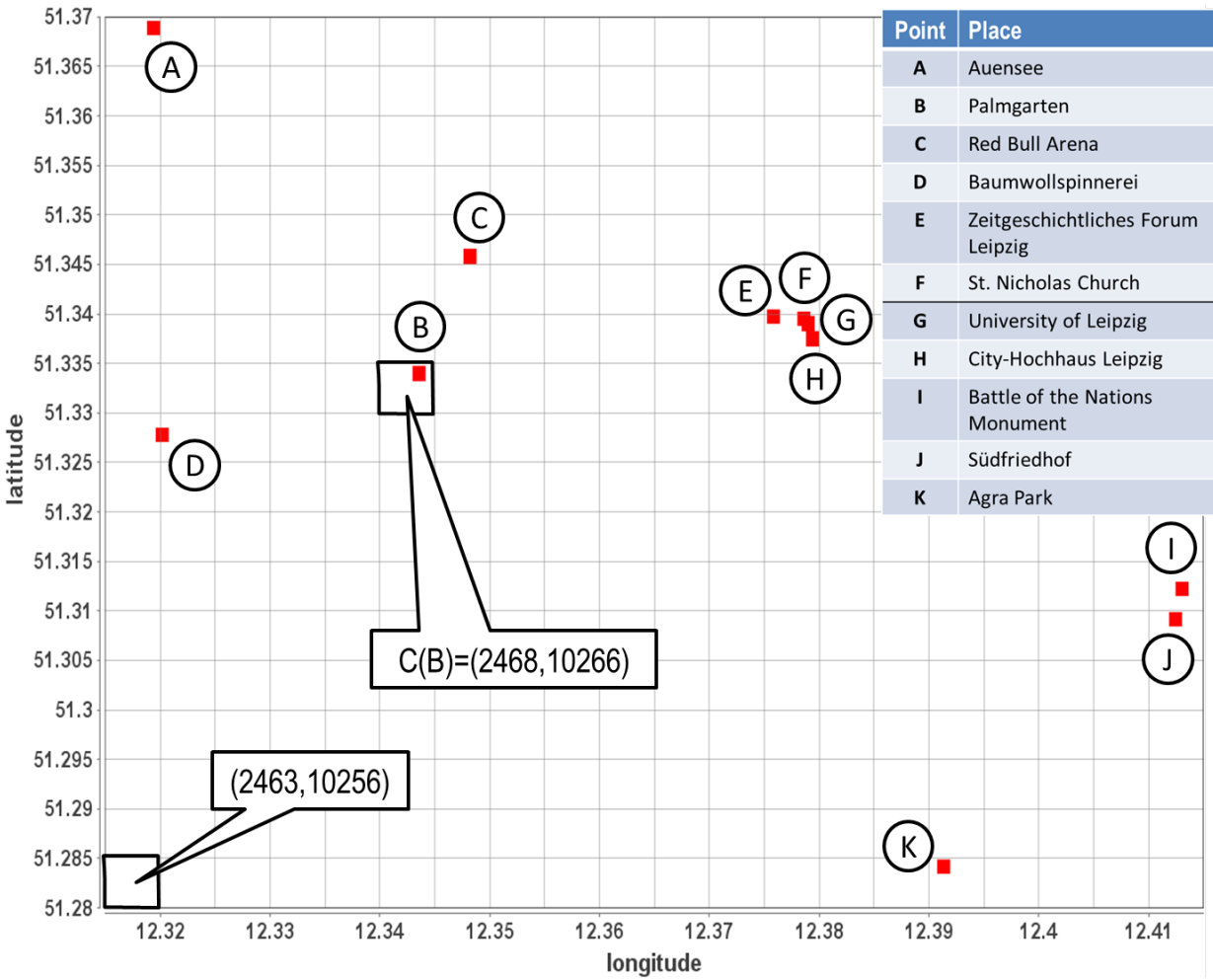
1. Assume spaces with Minkowski metric and $p \geq 2$

$$\delta(s,t) = \sqrt[p]{\sum_{i=1}^{n} |s_i - t_i|^p}$$



$\mathcal{HR}^3$

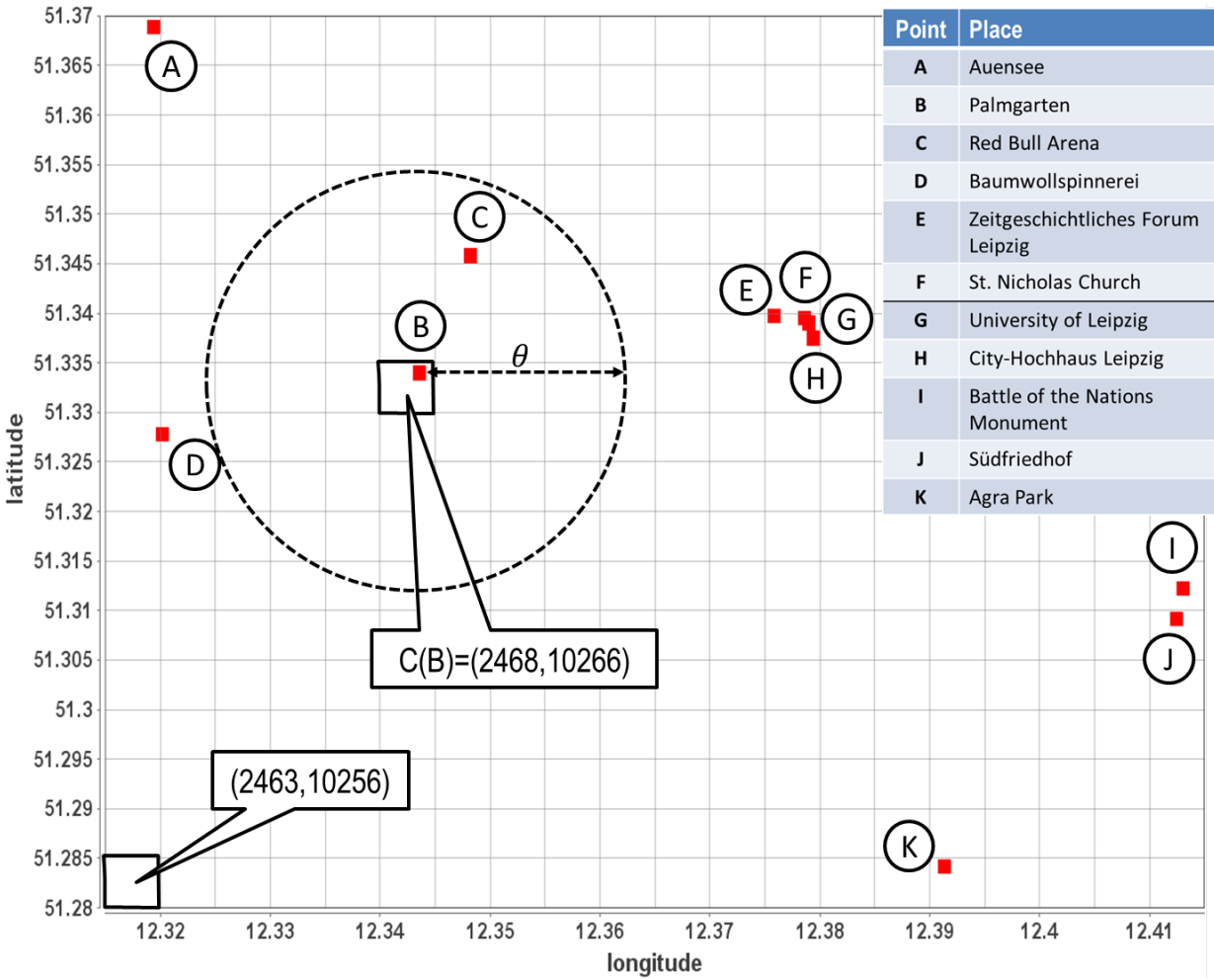
2. Create grid of width $\Delta = \theta/\alpha$



Point	Place	Longitude	Latitude	C(Point)	
A	Auensee	12.3194	51.3688	(2463,10273)	Π_0
B	Palmgarten	12.3436	51.3339	(2468,10226)	
C	Red Bull Arena	12.3482	51.3457	(2469,10269)	
D	Baumwollspinnerei	12.3201	51.3277	(2464,10265)	
E	Zeitgeschichtliches Forum Leipzig	12.3758	51.3397	(2475,10267)	
F	St. Nicholas Church	12.3786	51.3395	(2475,10267)	
G	University of Leipzig	12.3790	51.3390	(2475,10267)	Π_1
H	City-Hochhaus Leipzig	12.3794	51.3375	(2475,10267)	
I	Battle of the Nations Monument	12.4130	51.3122	(2482,10262)	
J	Südfriedhof	12.4125	51.3091	(2482,10261)	
K	Agra Park	12.3913	51.2841	(2478,10256)	

$\mathcal{HR}^3$

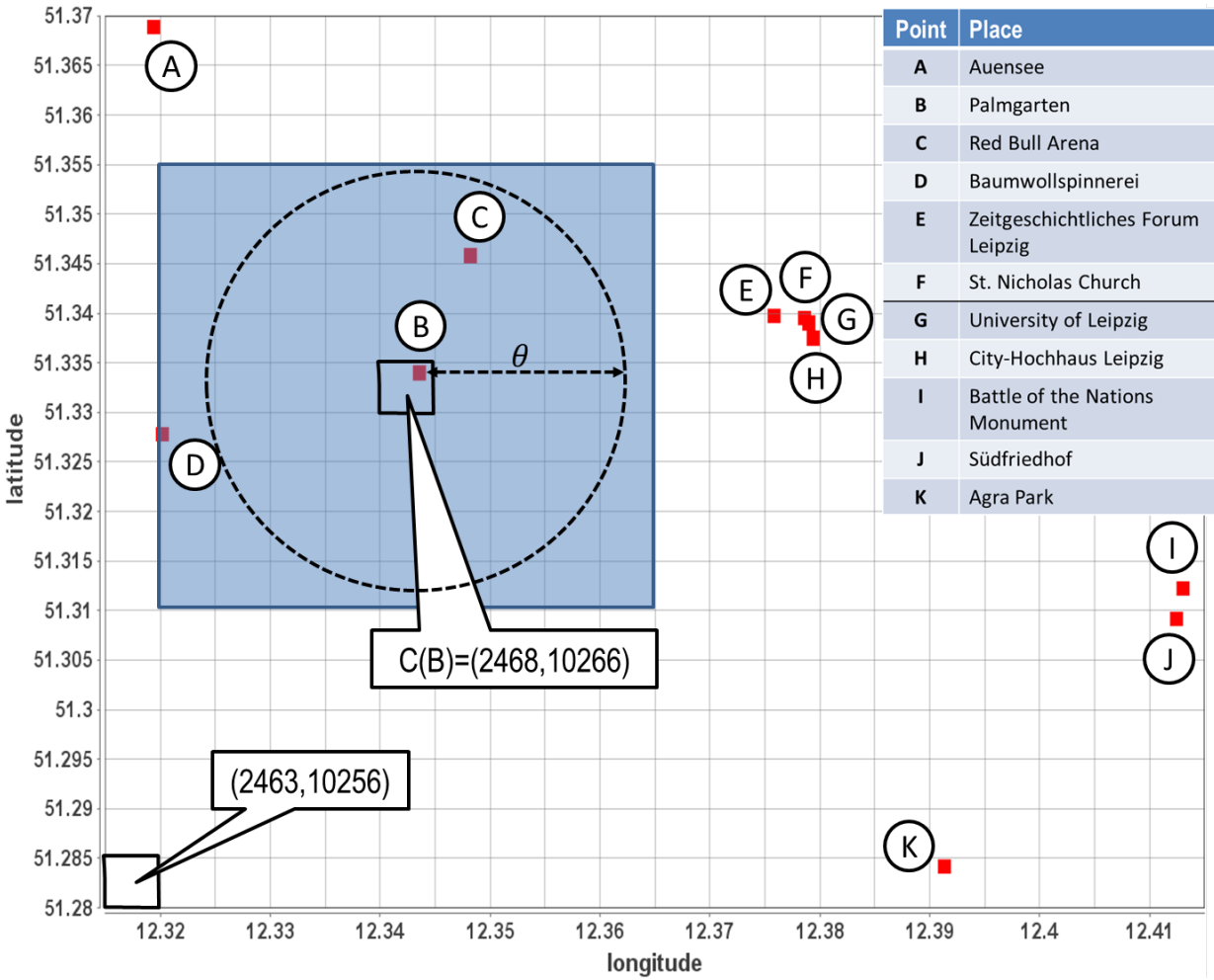
3. Link discovery condition describes a hypersphere



Point	Place	Longitude	Latitude	C(Point)	
A	Auensee	12.3194	51.3688	(2463,10273)	Π_0
B	Palmgarten	12.3436	51.3339	(2468,10226)	
C	Red Bull Arena	12.3482	51.3457	(2469,10269)	
D	Baumwollspinnerei	12.3201	51.3277	(2464,10265)	
E	Zeitgeschichtliches Forum Leipzig	12.3758	51.3397	(2475,10267)	
F	St. Nicholas Church	12.3786	51.3395	(2475,10267)	
G	University of Leipzig	12.3790	51.3390	(2475,10267)	Π_1
H	City-Hochhaus Leipzig	12.3794	51.3375	(2475,10267)	
I	Battle of the Nations Monument	12.4130	51.3122	(2482,10262)	
J	Südfriedhof	12.4125	51.3091	(2482,10261)	
K	Agra Park	12.3913	51.2841	(2478,10256)	

$\mathcal{HR}^3$

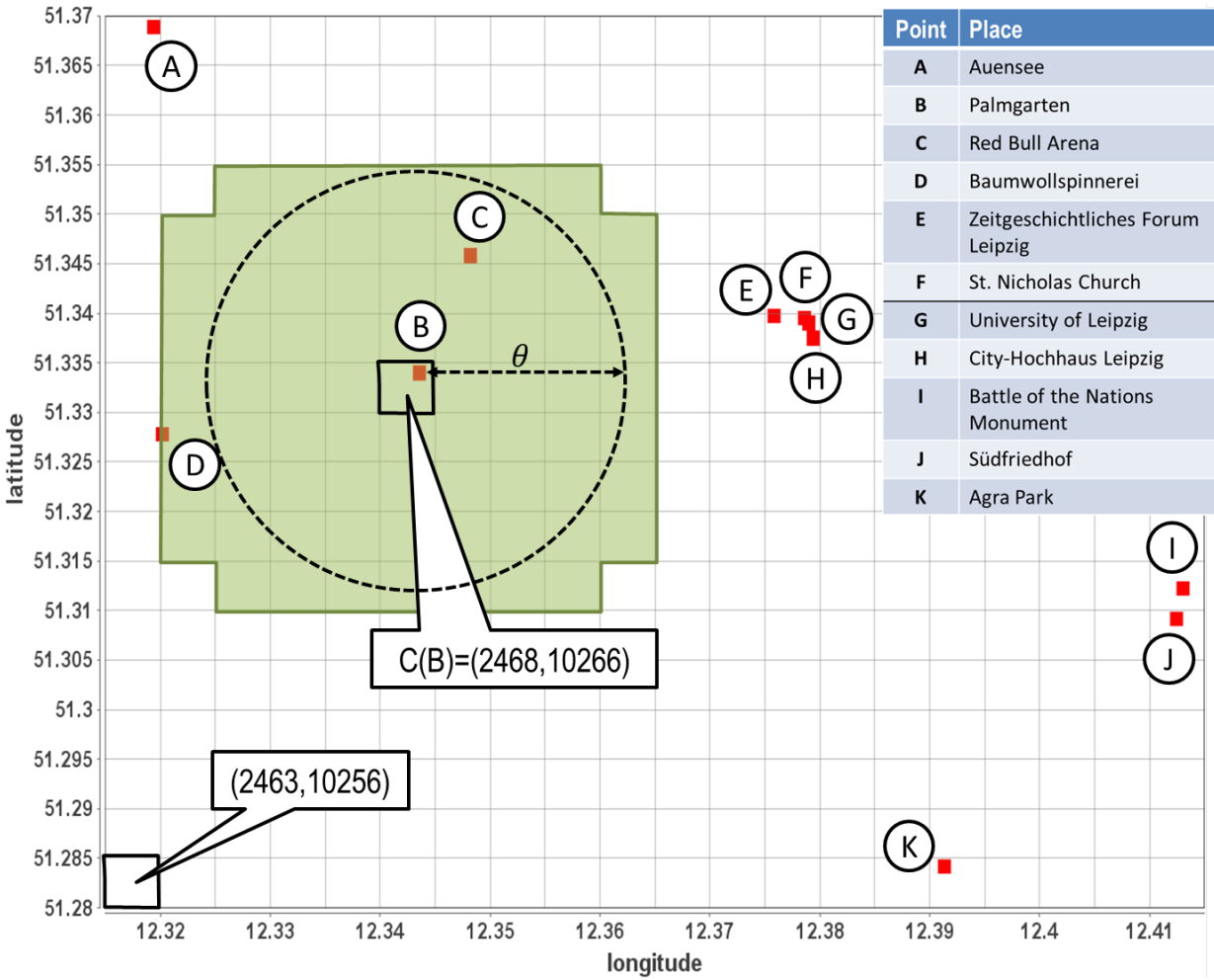
4 Approximate hypersphere with hypercube



Point	Place	Longitude	Latitude	C(Point)	
A	Auensee	12.3194	51.3688	(2463,10273)	Π_0
B	Palmgarten	12.3436	51.3339	(2468,10226)	
C	Red Bull Arena	12.3482	51.3457	(2469,10269)	
D	Baumwollspinnerei	12.3201	51.3277	(2464,10265)	
E	Zeitgeschichtliches Forum Leipzig	12.3758	51.3397	(2475,10267)	
F	St. Nicholas Church	12.3786	51.3395	(2475,10267)	
G	University of Leipzig	12.3790	51.3390	(2475,10267)	Π_1
H	City-Hochhaus Leipzig	12.3794	51.3375	(2475,10267)	
I	Battle of the Nations Monument	12.4130	51.3122	(2482,10262)	
J	Südfriedhof	12.4125	51.3091	(2482,10261)	
K	Agra Park	12.3913	51.2841	(2478,10256)	

$\mathcal{HR}^3$

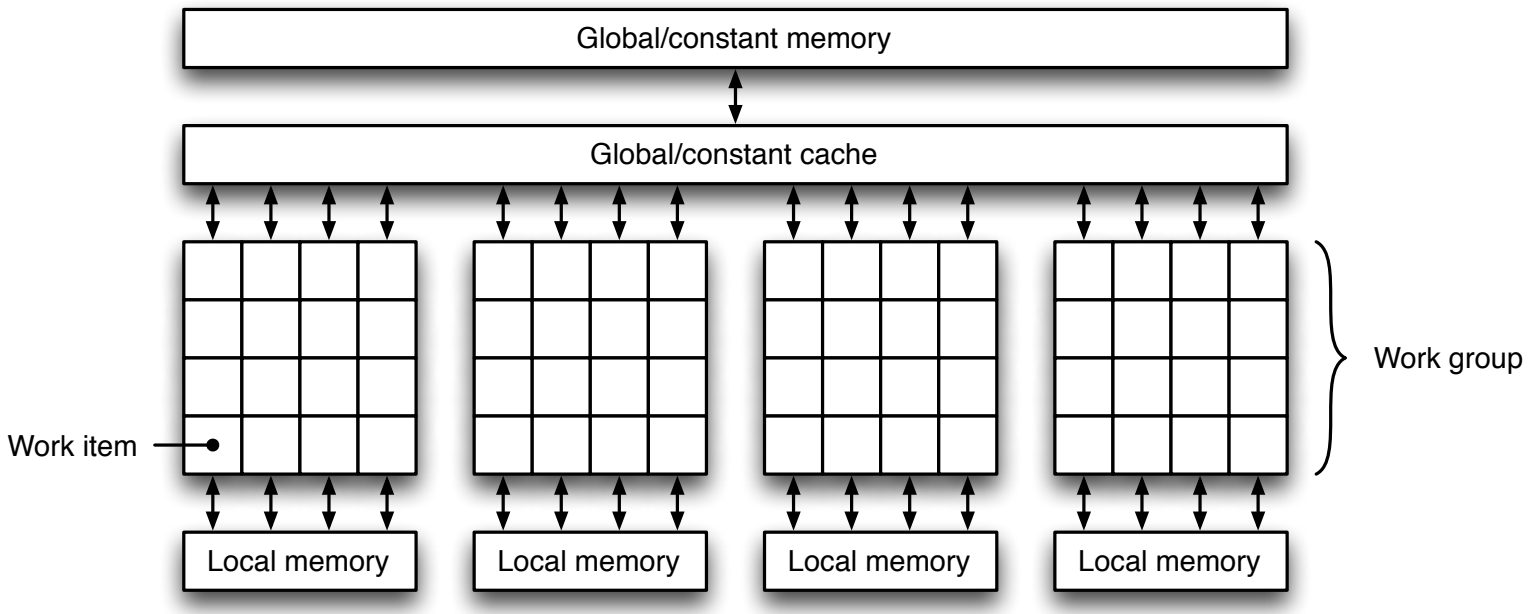
5. Use index to discard portions of hypercube



Point	Place	Longitude	Latitude	C(Point)	
A	Auensee	12.3194	51.3688	(2463,10273)	Π_0
B	Palmgarten	12.3436	51.3339	(2468,10226)	
C	Red Bull Arena	12.3482	51.3457	(2469,10269)	
D	Baumwollspinnerei	12.3201	51.3277	(2464,10265)	
E	Zeitgeschichtliches Forum Leipzig	12.3758	51.3397	(2475,10267)	
F	St. Nicholas Church	12.3786	51.3395	(2475,10267)	
G	University of Leipzig	12.3790	51.3390	(2475,10267)	Π_1
H	City-Hochhaus Leipzig	12.3794	51.3375	(2475,10267)	
I	Battle of the Nations Monument	12.4130	51.3122	(2482,10262)	
J	Südfriedhof	12.4125	51.3091	(2482,10261)	
K	Agra Park	12.3913	51.2841	(2478,10256)	

$\mathcal{HR}^3$ *in GPUs*

- Large number of simple compute cores
- Same instruction, multiple data
- Bottleneck: PCI Express Bus
  1. Run discretization on CPU
  2. Run indexing on GPU
  3. Run comparisons on CPU

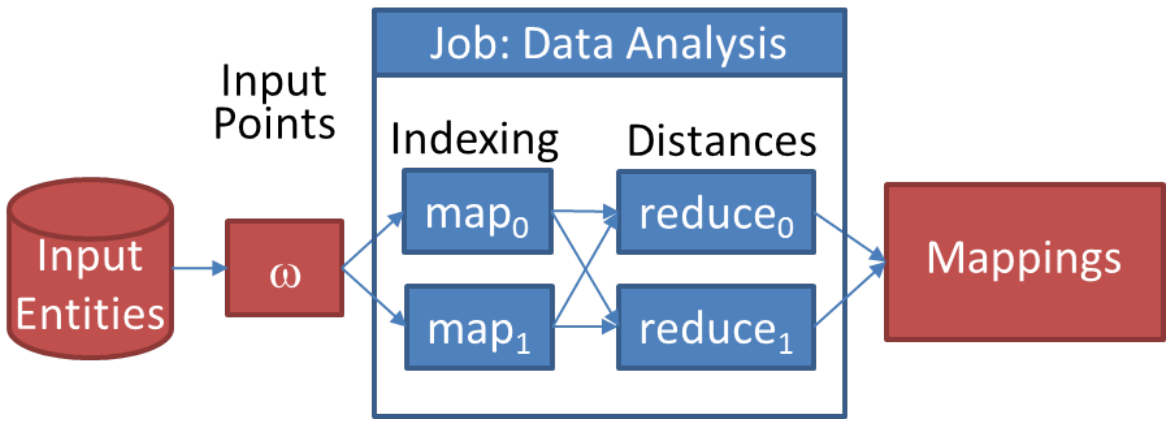


$\mathcal{HR}^3$ *on the Cloud*

- **Naive Approach**

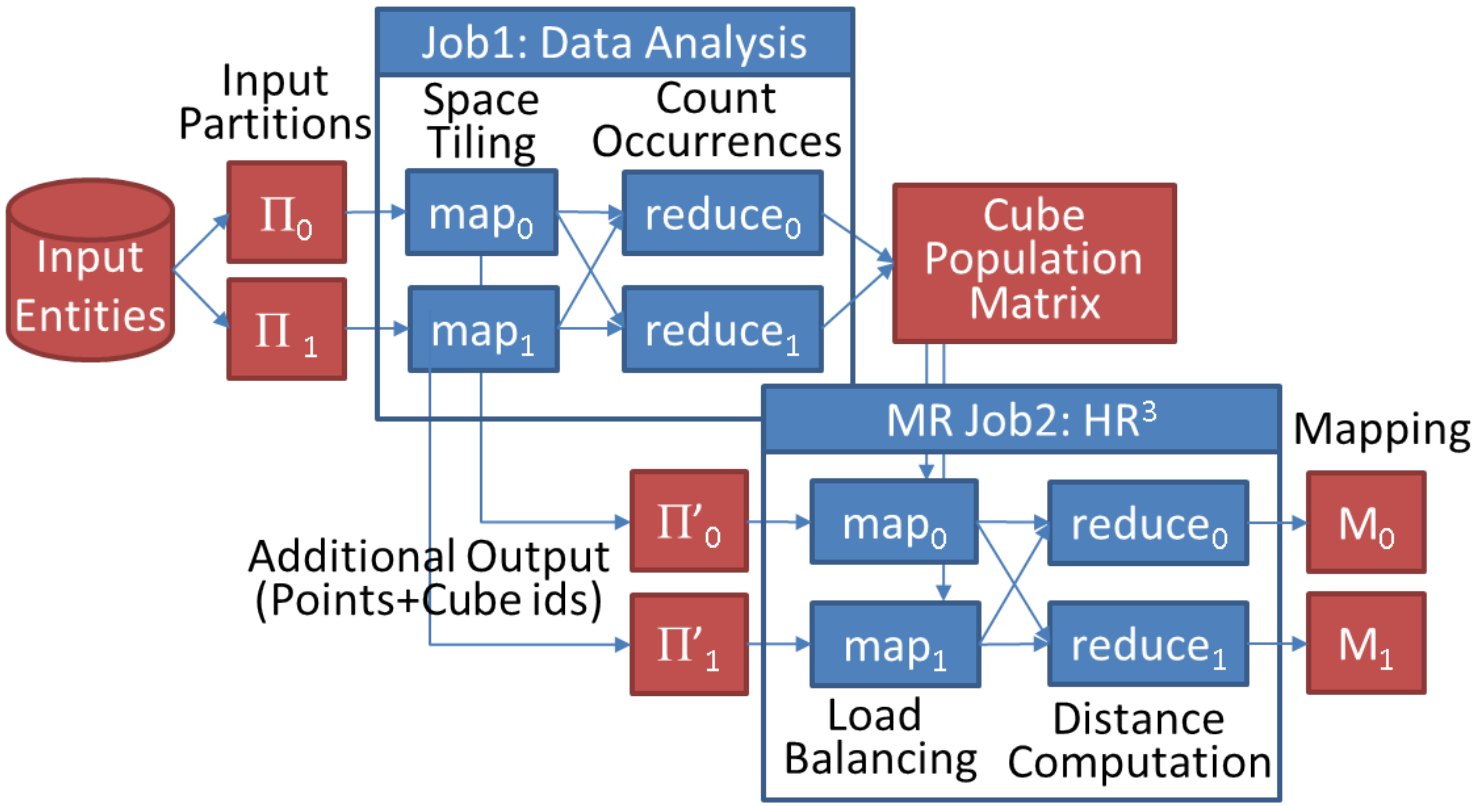
$\mathcal{HR}^3$ *on the Cloud*

- **Naive Approach**
  1. Rely on Map Reduce paradigm
  2. Run discretization and assignment to cubes in map step
  3. Run distance computation in reduce step



$\mathcal{HR}^3$ *on the Cloud*

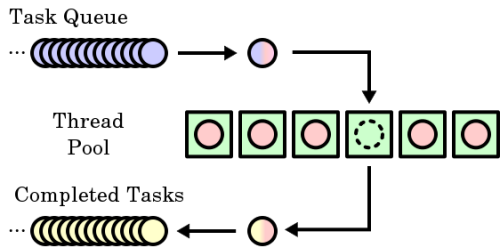
- **Load Balancing**
  1. Run two jobs
  2. Job1: Compute cube population matrix
  3. Job2: Distribute balanced linking tasks across mappers and reducers

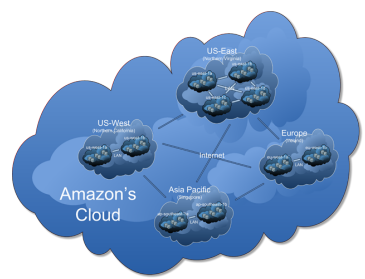


Outline

Hardware

1. CPU (Java)
   - 32-core server running Linux 10.0.4
   - AMD Opteron 6128 clocket at 2.0GHz
2. GPU (C++)
   - AMD Radeon 7870 with 20 compute units, 64 parallel threads
   - Host program ran on Intel Core i7 3770 CPU with 8GB RAM and Linux 12.10
   - Ran the Java code on the same machine for scaling
3. Cloud (Java)
   - 10 c1.medium nodes (2 cores, 1.7GB) for small experiments
   - 20 c1.large nodes (8 cores, 7GB) for large experiments





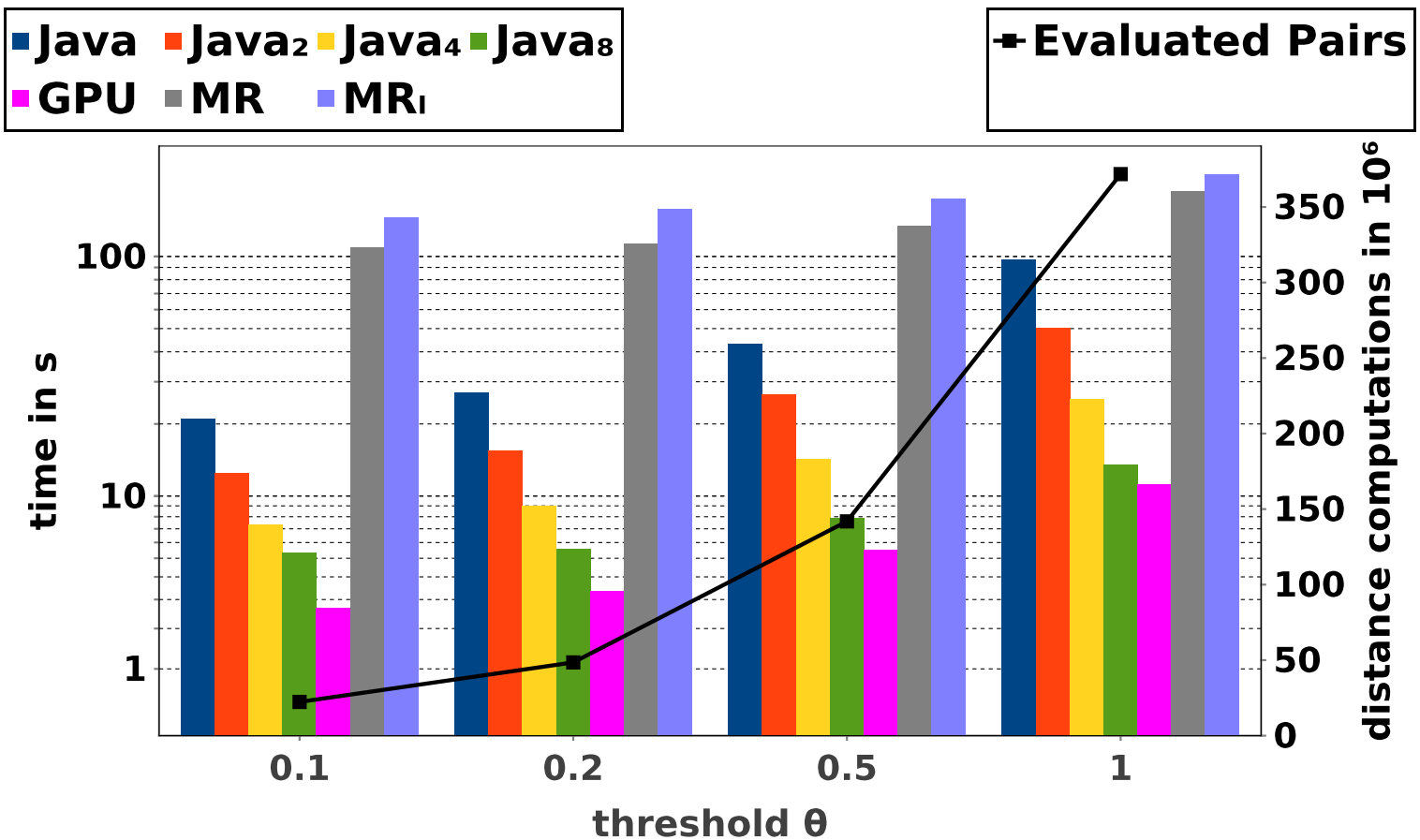
Experimental Setup

- Run deduplication task
- Evaluate behavior on different number of dimensions
- **Important**: Scale results
  - Different hardware (2-7 times faster C++ workstation)
  - Programming language
- Evaluate scalability (DS_4)

Dataset	Source	Size	Features
DS_1	DBPedia	25,781	min/medium/max elevation
DS_2	DBPedia	475,000	latitude, longitude
DS_3	Linked Geo Data	500,000	latitude, longitude
DS_4	Linked Geo Data	6,000,000	latitude, longitude

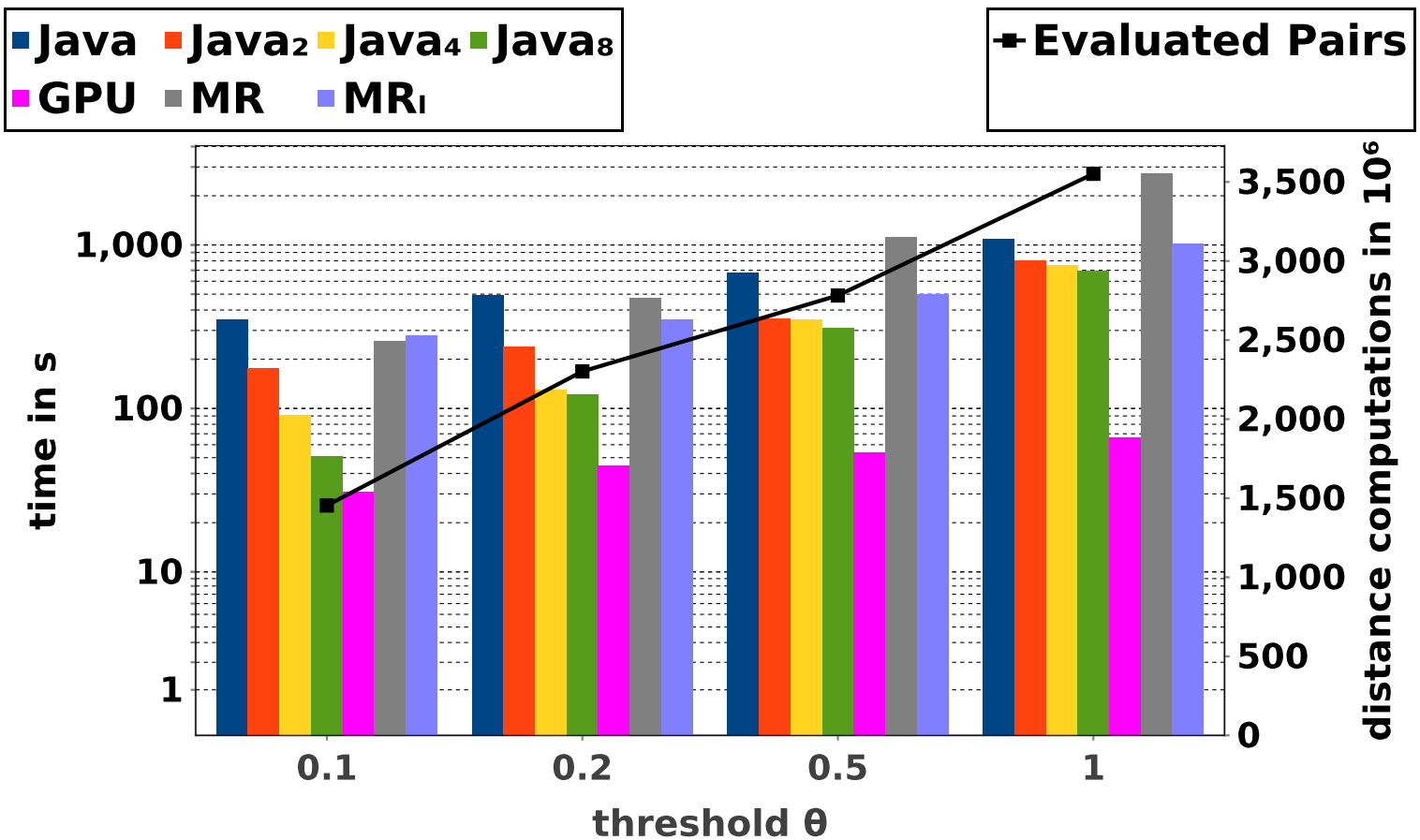
Results – DS1

- DBpedia, 3 dimensions, 26K
- CPUs scale better across different θ



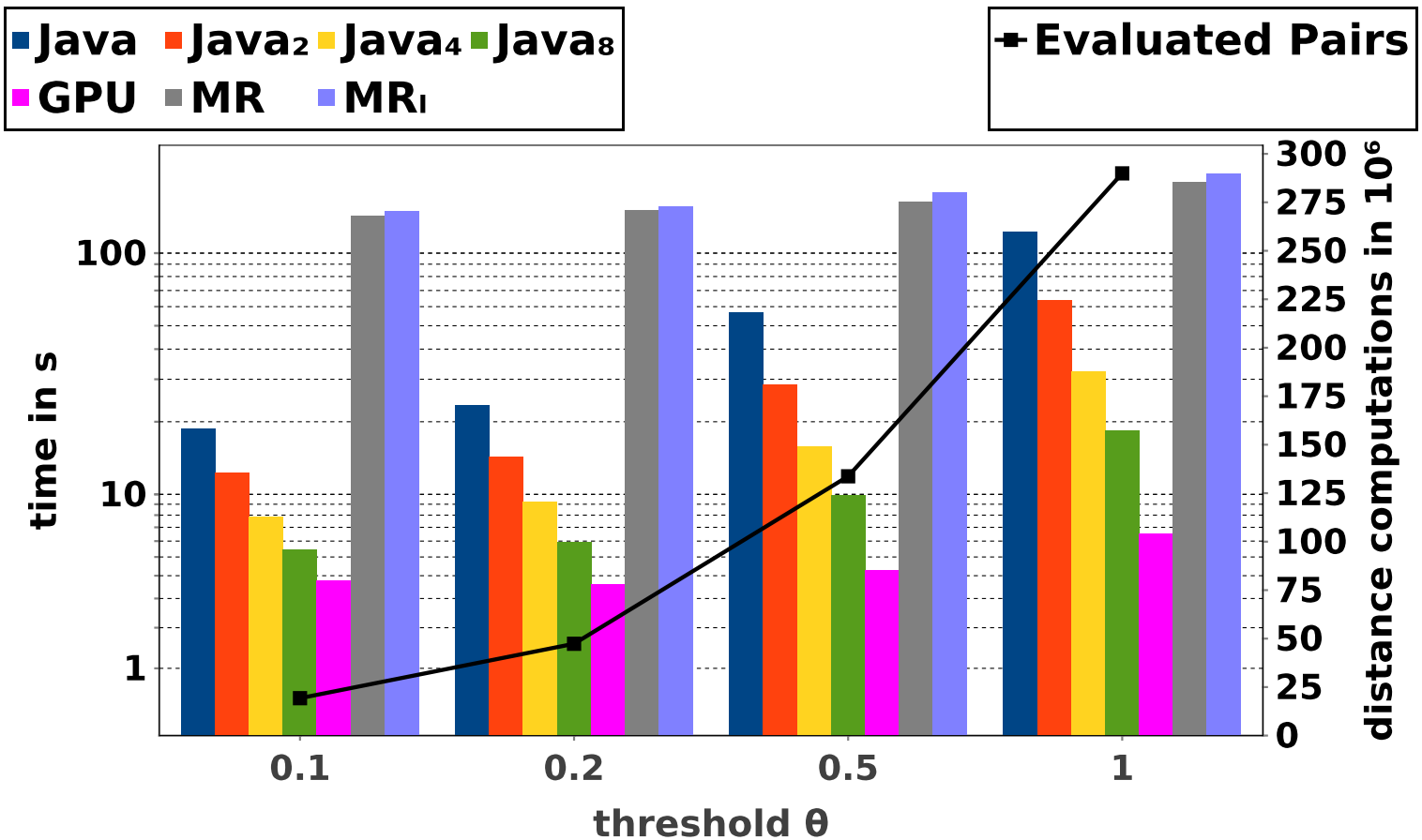
Results – DS2

- DBpedia, 2 dimensions, 475K
- GPUs scale better across different θ
- **Break-even point** $\approx 10^8$ results



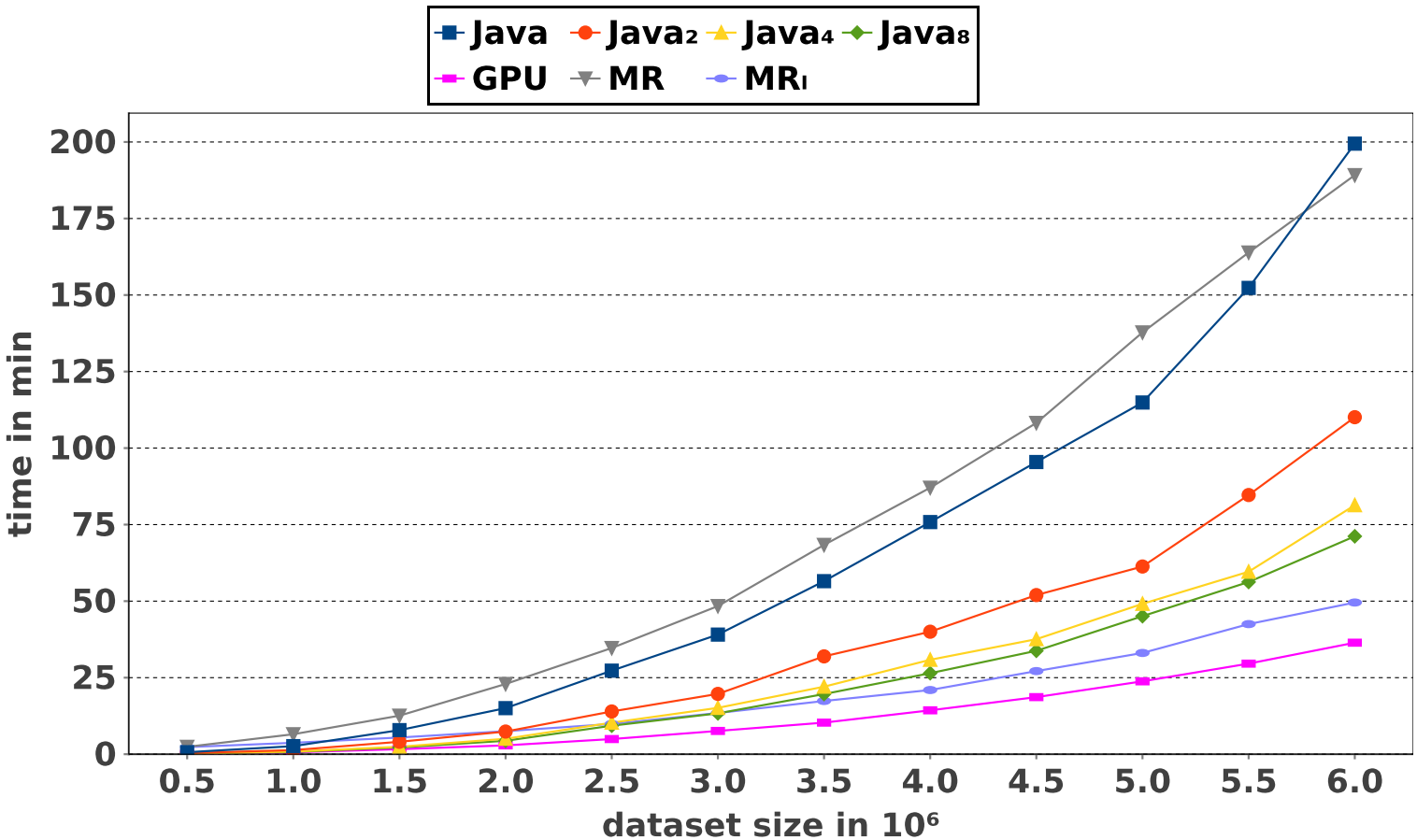
Results – DS3

- LinkedGeoData, 2 dimensions, 500K
- Similar picture



Results – Scalability

- LinkedGeoData, 2 dimensions, 6M
- Cloud (with load balancing) better for $\approx 10^{10}+$ results with 30 nodes



Outline

Summary

- **Question**
  - When should we use which hardware for link discovery?
- **Results**
  - Implemented $\mathcal{HR}^3$ on different hardware
  - Provided the first implementation of link discovery on GPUs
  - Devised a load balancing approach for linking on the cloud
  - Discovered 10^8 and 10^{10} results as break-even points

Summary

- **Question**
  - When should we use which hardware for link discovery?
- **Insights**
  - New hardware requirements for Big Data Processing
  - E.g., GPUs with buses faster than PCIe (e.g., Firewire speed)
  - **Accurate use of local resources sufficient for most of the current applications**

Vision

- **Self-adaptive link discovery frameworks**
  1. Check for free hardware
  2. Use free resources to be time-efficient
- Already developed approaches for predicting
  1. Result size of algorithms
  2. Runtimes on different hardware
- Need to develop **orchestration approach**

Thank You!

Questions?

Axel Ngonga
Augustusplatz 10
D-04109 Leipzig
ngonga@informatik.uni-leipzig.de
http://aksw.org/AxelNgonga
http://limes.sf.net