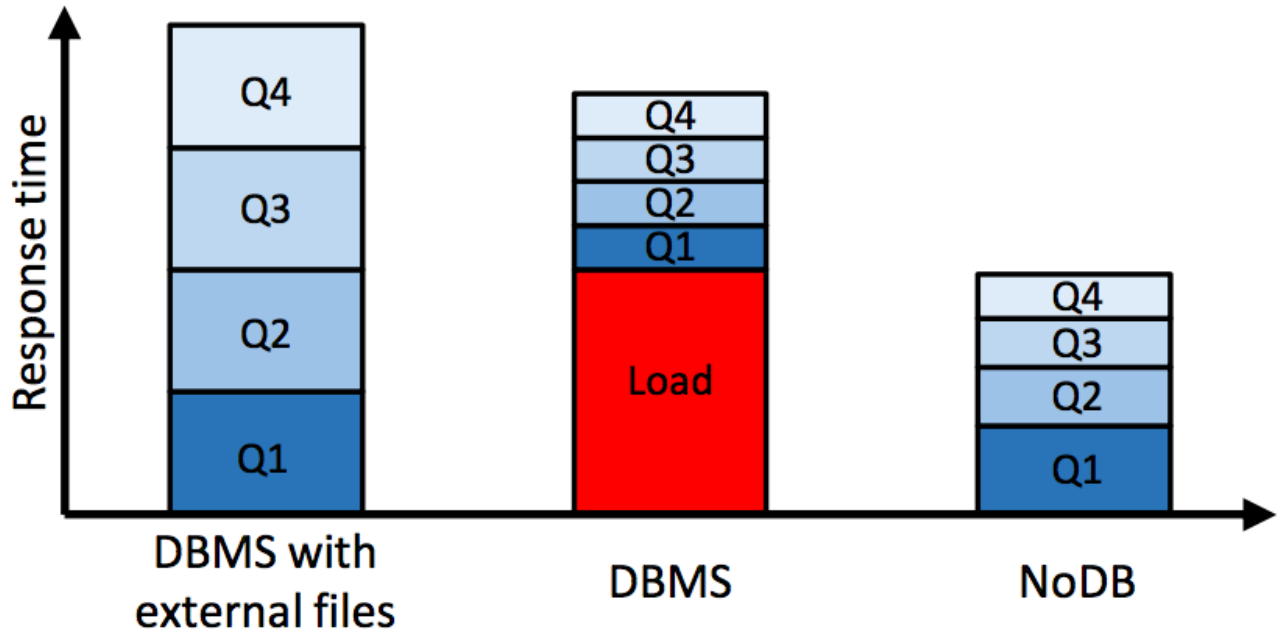
NoDB: Efficient Query Execution on Raw Data Files*

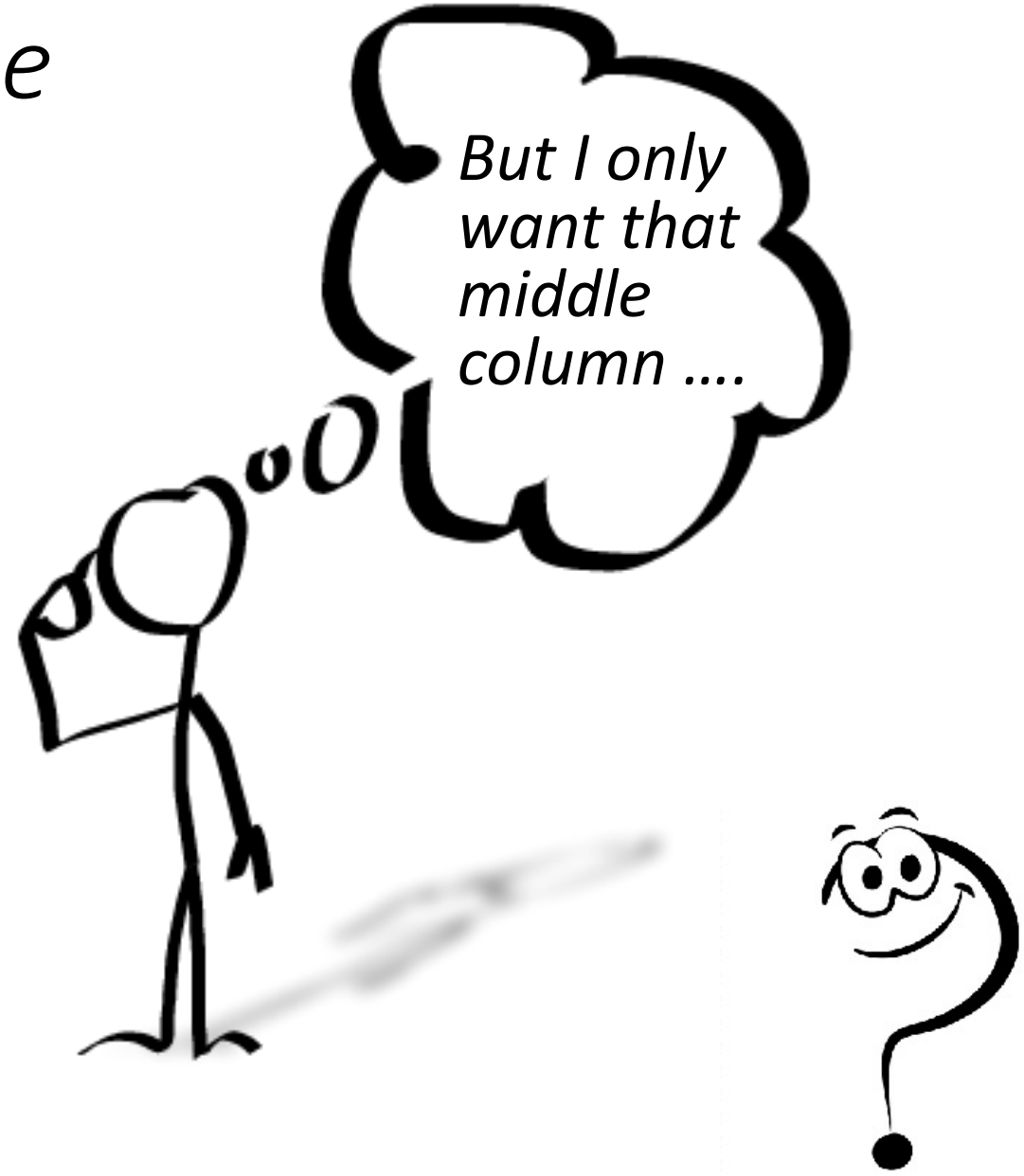
Ioannis Alagiannis† Renata Borovica-Gajic† Miguel Branco† Stratos Idreos‡ Anastasia Ailamaki†

†Ecole Polytechnique Fédérale de Lausanne
{ioannis.alagiannis, renata.borovica, miguel.branco, anastasia.ailamaki}@epfl.ch

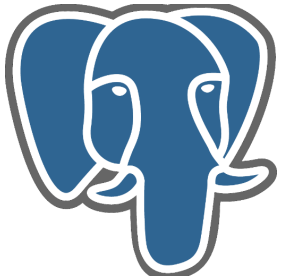
‡Harvard University
stratos@seas.harvard.edu



Problem - The *Data Deluge*



Important/hard? - Emerging Applications



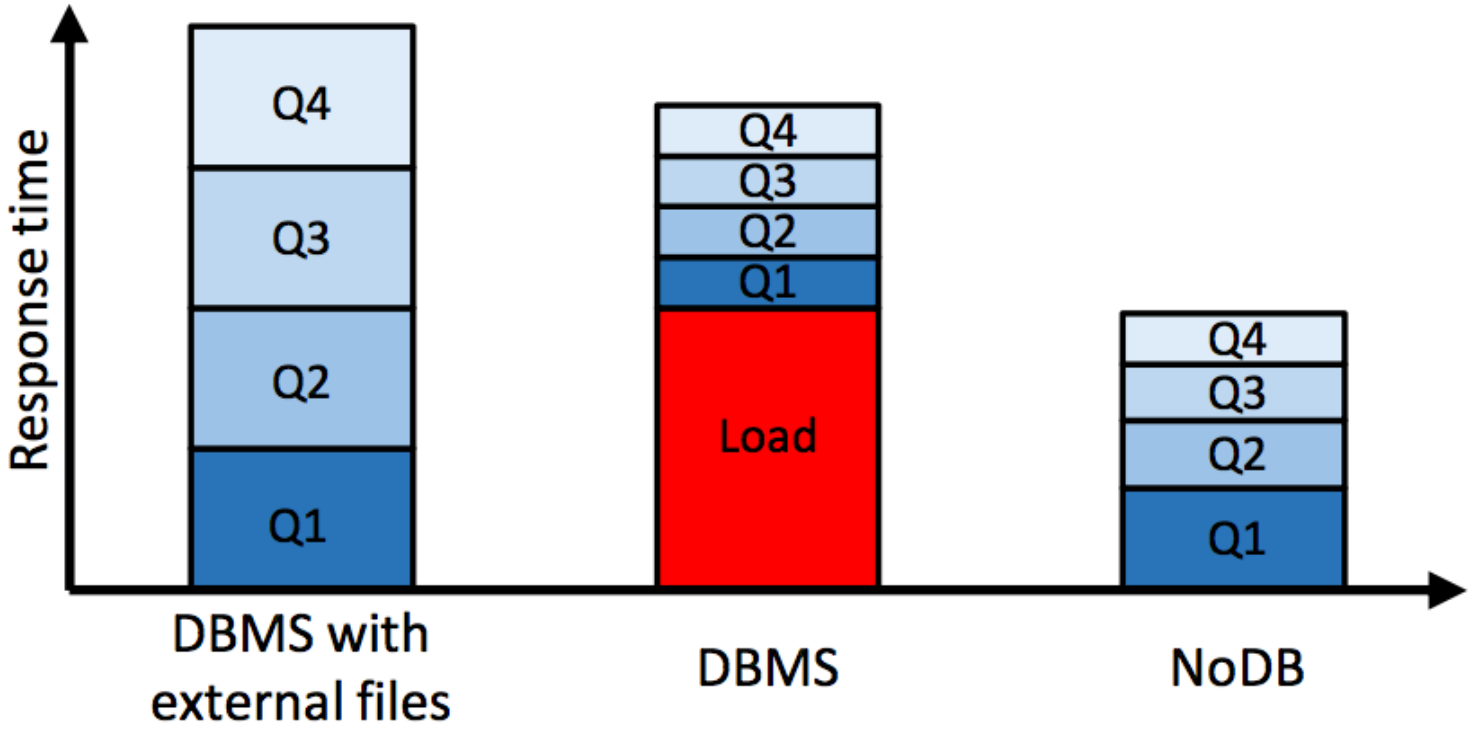
ORACLE
DATABASE

OR

High initial cost

High complexity,
Lack of features

The NoDB Philosophy



Straightforward Approaches

Query data.csv
->Load(“data.csv”)
Query data.csv
…

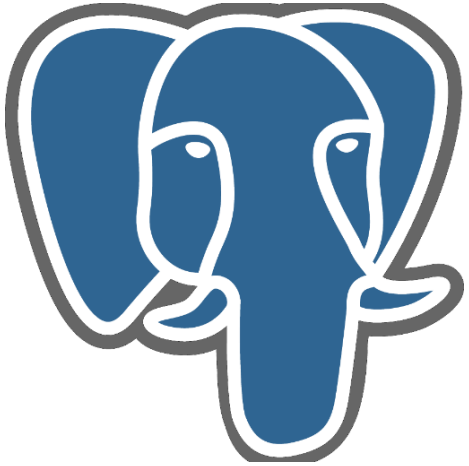
Query moredata.csv
->Load(“moredata.csv”)
Query data.csv
Query moredata.csv
…

Straightforward Approaches

Query data.csv
->Scan(“data.csv”)
Query data.csv
->Scan(“data.csv”)
…

Query moredata.csv
->Scan(“moredata.csv”)
Query data.csv
->Scan(“data.csv”)
Query moredata.csv
->Scan(“moredata.csv”)
…

PostgresRaw



+ NoDB

On-the-fly Parsing – Selective Tokenizing

E for B < 50 && D < 100

A, B, C, D, E, …

…

“100”, “55”, “13”, ”44”, ”37”, …

…

“123”, “23”, “12”, “78”, “2”, …

E for B < 50 && D < 100

Scan row for attributes

“55”, “44”, “37”, …

Stop after scanning E

Move on to next row

…

On-the-fly Parsing – Selective Parsing

E for B < 50 && D < 100

A, B, C, D, E, …

…

“100”, “55”, “13”, ”44”, ”37”, …

…

“123”, “23”, “12”, “78”, “2”, …

E for B < 50 && D < 100

Begin to parse attributes

55, “44”, “37”, …

Stop after parsing B

Move on to next row

…

On-the-fly Parsing – Selective Tuple Formation

E for B < 50 && D < 100

A, B, C, D, E, …

...

“100”, “55”, “13”, ”44”, ”37”, ...

...

“123”, “23”, “12”, “78”, “2”, ...

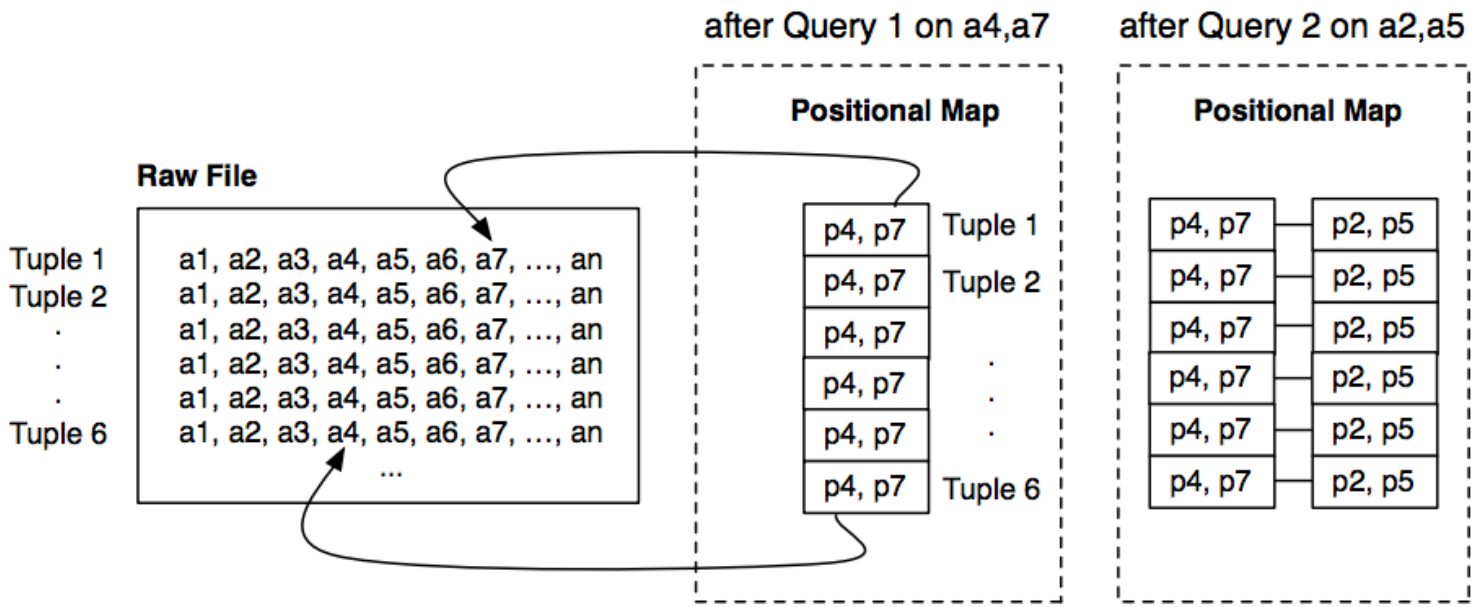
E for B < 50 && D < 100

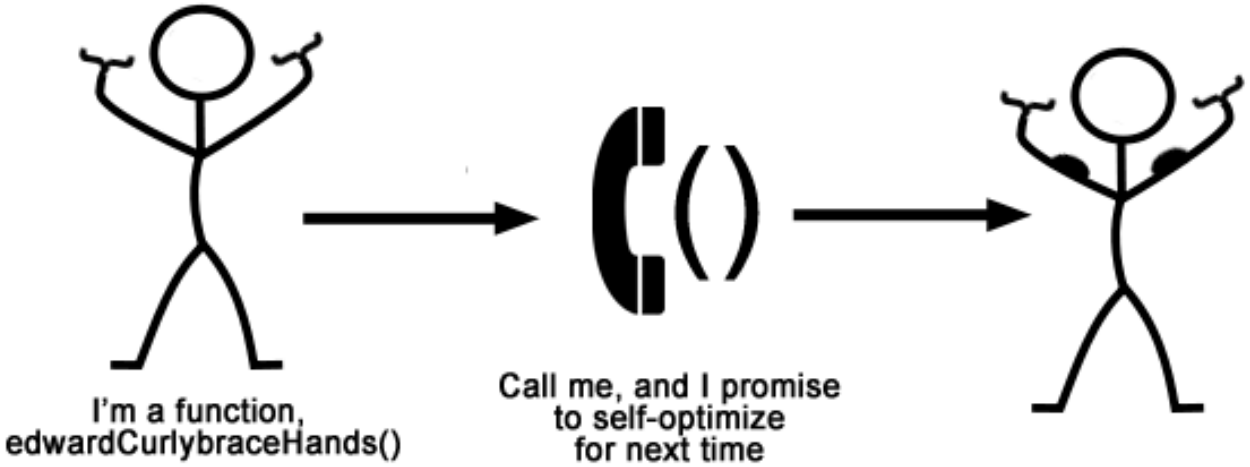
Form tuples of only qualifying rows, containing only required attributes

(23, 78, 2)

...

Adaptive Positional Map





Caching + Statistics

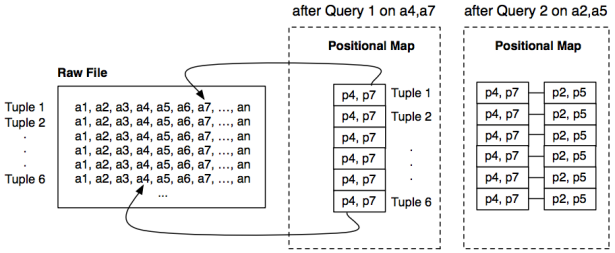
Which features address which strawman problems?

Which features address which strawman problems?

Loading Overheads

Parsing

Indexing



Statistics

Experiments – Indexing (Positional Map)

Random queries accessing 10(out of 150) attributes
Vary positional map size from 14.3MB to 2.1GB

Queries quickly take advantage of information added to positional map

File size gradually increased from 2GB to 92GB

Unlimited positional map size

Observe linear scalability

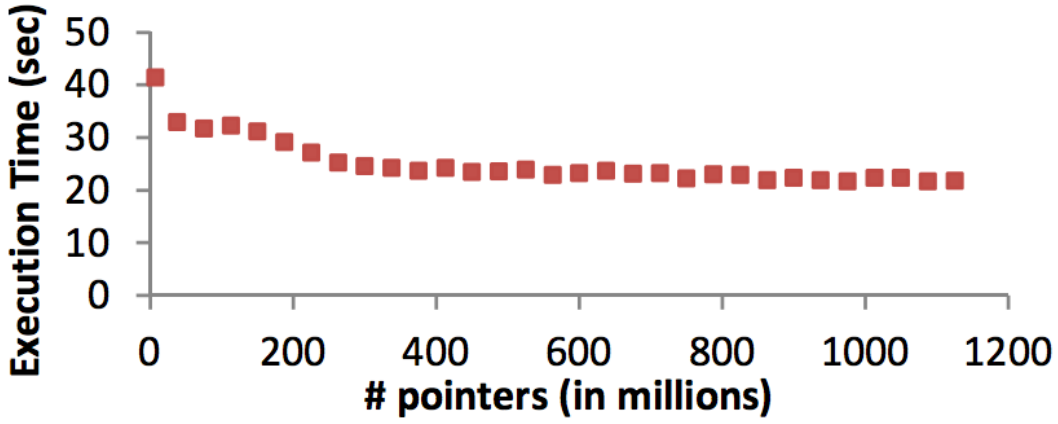


Figure 3: Effect of the number of pointers in the positional map.

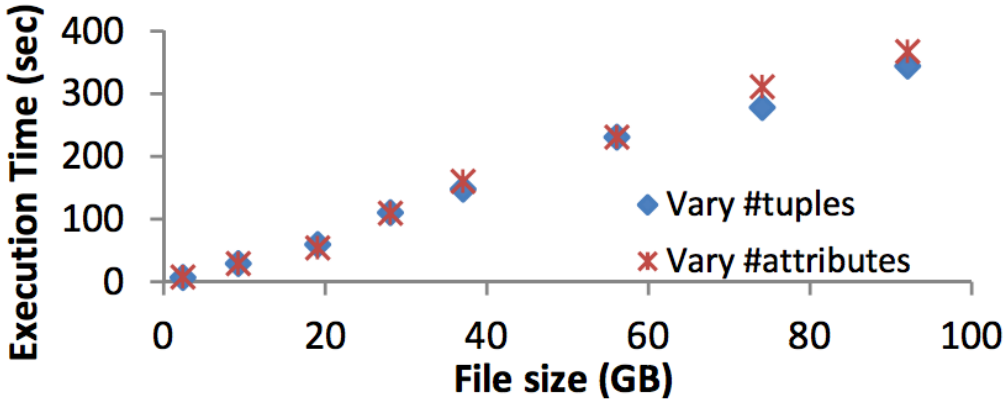


Figure 4: Scalability of the positional map.

Experiments – Positional Map + Caching

Random queries accessing 5(out of 150) attributes

Positional map provides fairly consistent parsing speedup

Caching provides maximum benefits when all attributes are cached

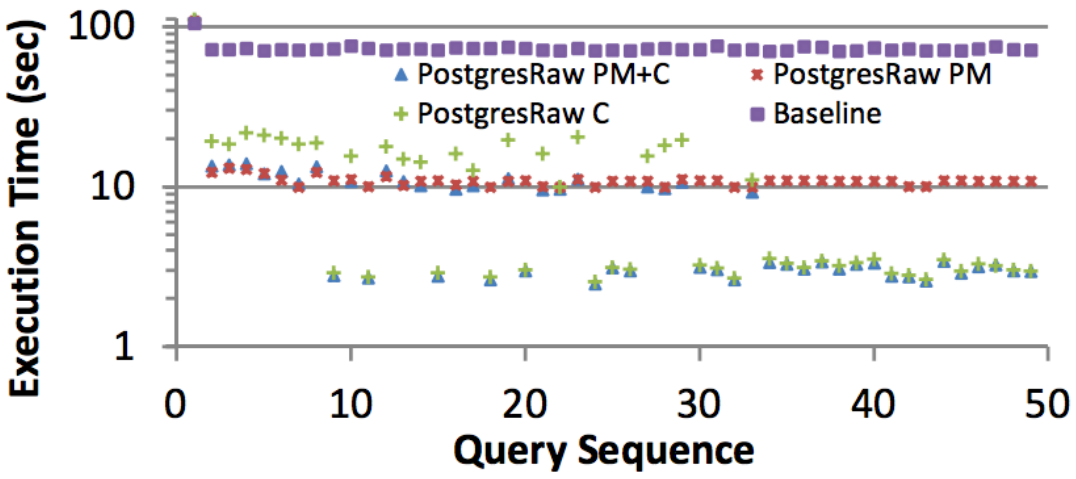


Figure 5: Effect of the positional map and caching.

Experiments – Adaptivity

Epochs of 50 queries accessing a subset of the attributes randomly
1 – unique subset, 2 – unique subset
3 – from both 1 & 2, 4 – overlaps with 2, 5 – slight shift from 4

PosgresRaw quickly adapts to workload changes

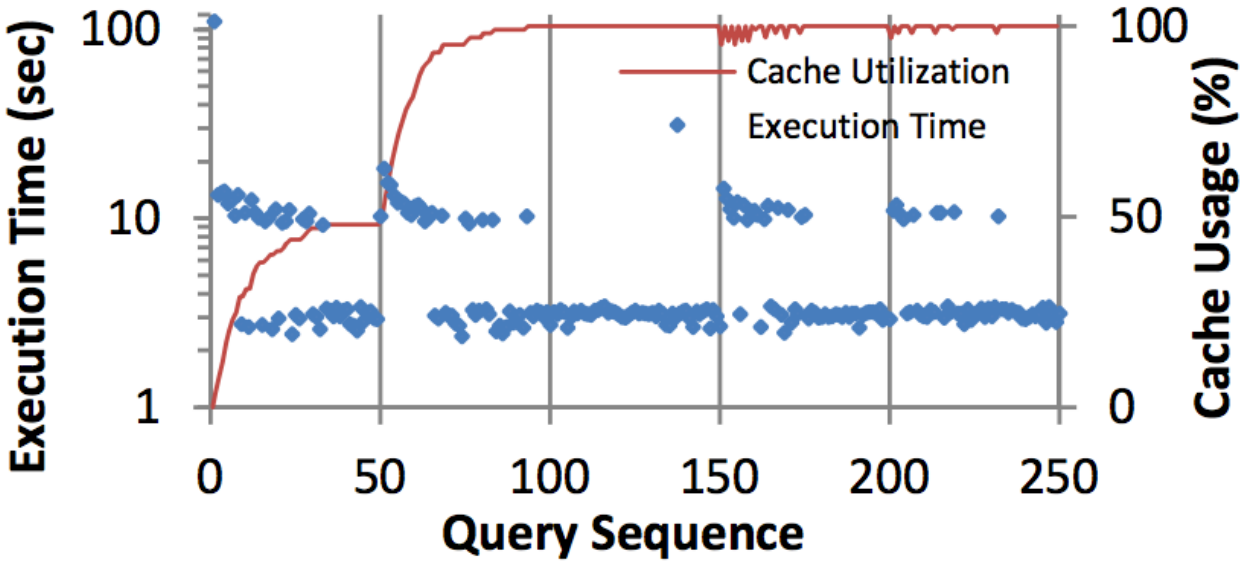


Figure 6: Adapting to changes in the workload.

Experiments – Statistics

PosgresRaw w/ statistics enabled and w/o

Statistics gathering adds minimal overhead to scanning, with large benefits to query planning

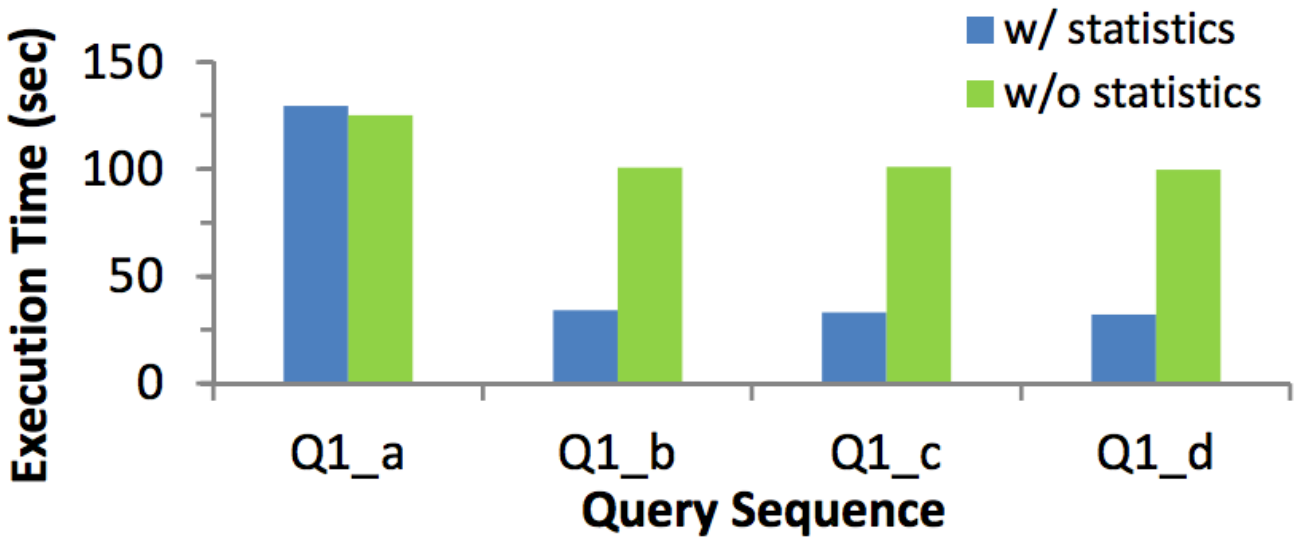
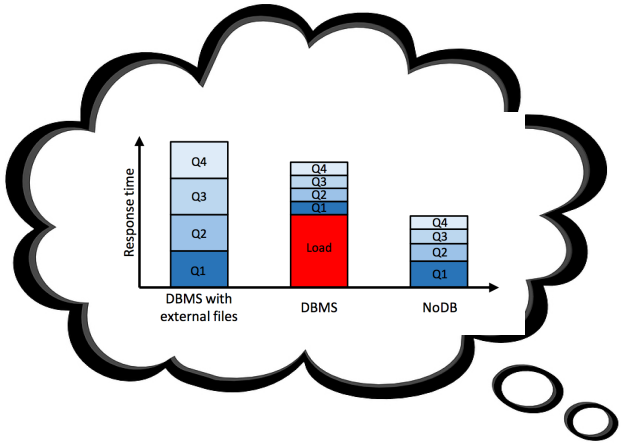


Figure 8: Execution time as PostgresRaw generates statistics.

Experiments – RDBMS vs. NoDB

Claims Proven?

Start accessing all data,
then access fewer rows,
then fewer columns



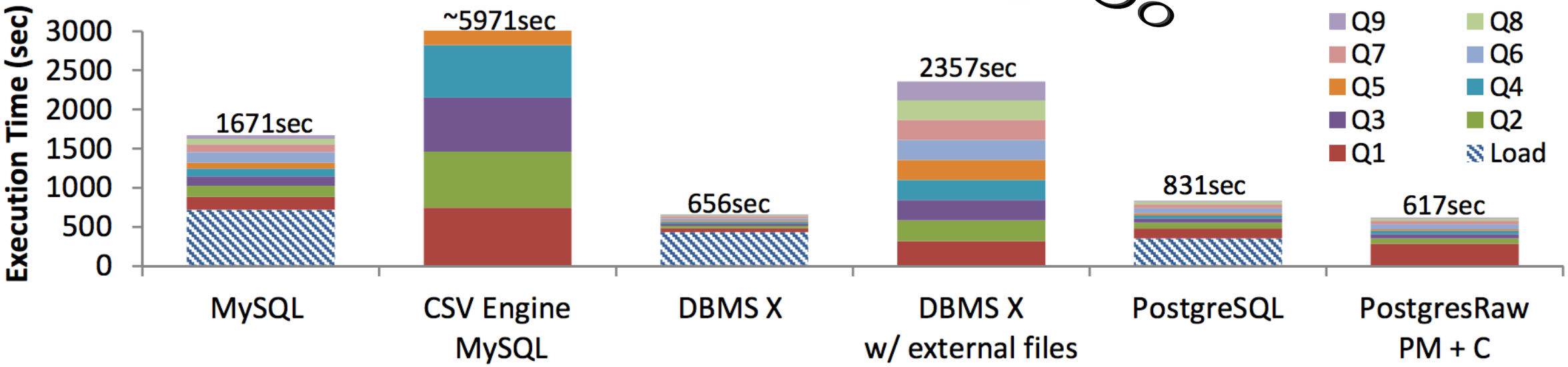


Figure 7: Comparing the performance of PostgresRaw with other DBMS.

Next steps

Flexible Storage/ Information Integration – PolyDB

Adaptive Indexing

Exploit some idle-time budget for auto-tuning

Exploit file-system interface – intercept syscalls to cache data