MonetDB in the context of the new (high-cadence) astronomical observatories

Bart Scheers

Centrum Wiskunde & Informatica, Amsterdam
NWO Dome: Big Bang, Big Data; P2: Access Patterns

1st Asterics–Obelics Workshop
Rome, 12–14 December 2016

M. Kersten, H. Mühleisen (CWI),
P. Groot, S. Bloemen (RU), P. Schellart (RU, Princeton),
A. van Elteren (Leiden)

H2020-Astronomy ESFRI and Research Infrastructure Cluster
(Grant Agreement number: 653477).

Historical Background

1975 — Row stores architecture standard for Relational DBMS, but column-oriented systems exist using transposed file approach

1980+ — IBM DB2, MS SQL Server, Oracle, ...
Transactional workloads (OLTP, row-wise access)
I/O based processing

1985 — **Decomposition Storage Model** (DSM), by Copeland & Khoshafian, 1985. Stimulates more research (on relational operators, e.g., joins, projection indices)

Storage Models – Row Stores/NSM

people

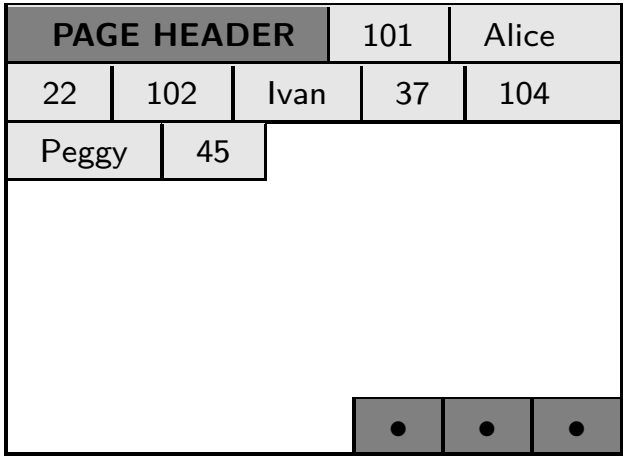
(void)	id	name	age
0	101	Alice	22
1	102	Ivan	37
2	104	Peggy	45
3	105	Victor	25
4	108	Eve	19
5	109	Walter	31
6	112	Trudy	27
7	113	Bob	29
8	114	Zoe	42
9	115	Charlie	35

Storage Models – Row Stores/NSM

people

(void)	id	name	age
0	101	Alice	22
1	102	Ivan	37
2	104	Peggy	45
3	105	Victor	25
4	108	Eve	19
5	109	Walter	31
6	112	Trudy	27
7	113	Bob	29
8	114	Zoe	42
9	115	Charlie	35

Row-Storage Model



Storage Models – Column Stores/DSM

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

Storage Models – Column Stores/DSM

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

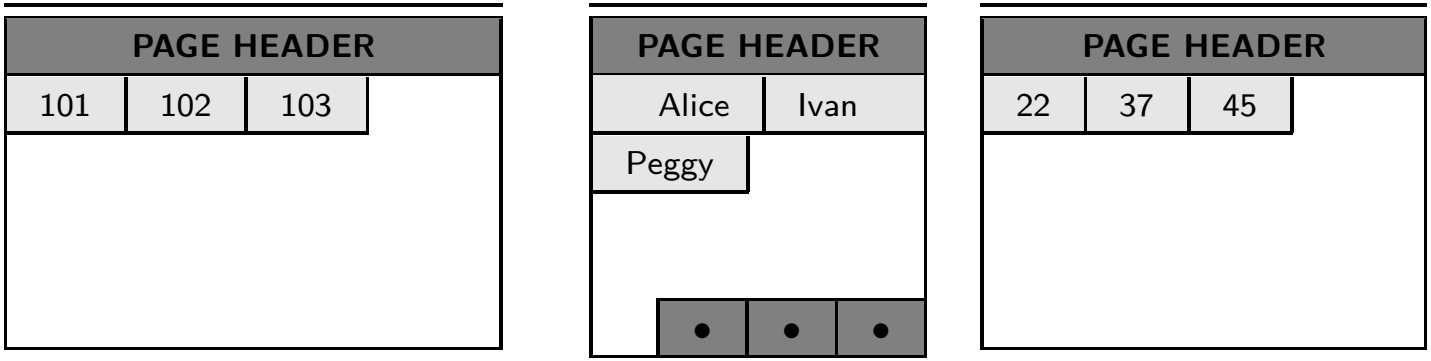
people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

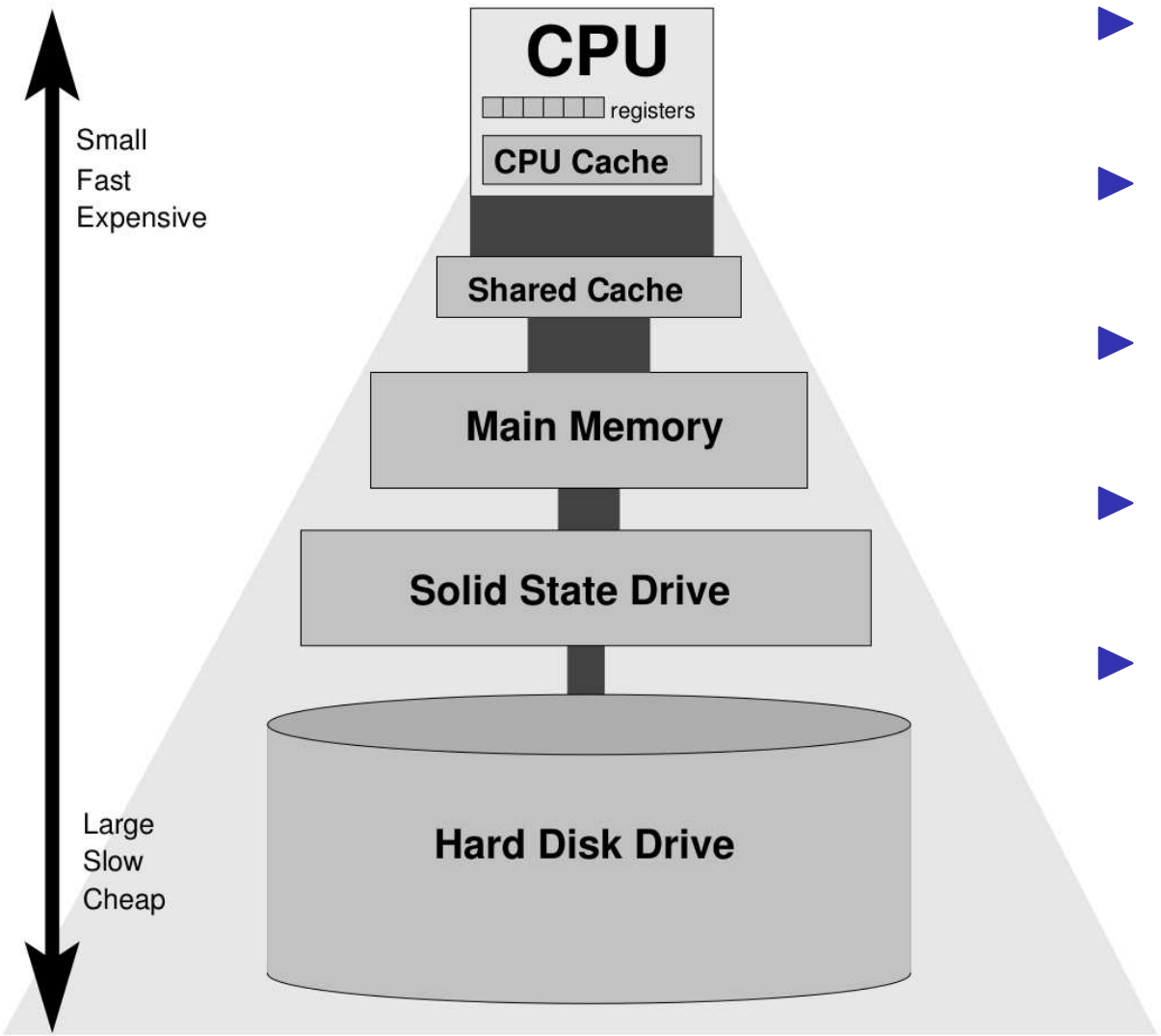
Columnar Storage Model



Historical Background

1975	Row stores architecture standard for Relational DBMS, but column-oriented systems exist using transposed file approach
1980+	IBM DB2, MS SQL Server, Oracle, ... Transactional workloads (OLTP, row-wise access) I/O based processing
1985	**Decomposition Storage Model** (DSM), by Copeland & Khoshafian, 1985. Stimulates more research (on relational operators, e.g., joins, projection indices)
1990–	Workload changes: OLTP vs. OLAP, BI, Data mining, ...
–2000	Hardware changes $\Rightarrow$ CPU is 60–90% idle, waiting for memory $\Rightarrow$ Databases hit The Memory Wall

Hardware changes: The Memory Hierarchy



- Data has to pass all layers to CPU
- Data closest to CPU fastest accessible
- CPU latency grown to 100–1000s of cycles
- Avoid random I/O access
- Read large multi-block sequences

Historical Background

2000+	Internet-scale applications collect unprecedented volumes of data for analysis and multi-terabyte data warehousing leading to revisting column-store technologies
2002	MonetDB goes open-source
–2010	SSDs
2013	Many new column-oriented DBMSs Traditional vendors added features or acquired systems

Historical Background

2000+	Internet-scale applications collect unprecedented volumes of data for analysis and multi-terabyte data warehousing leading to revisting column-store technologies
2002	MonetDB goes open-source
–2010	SSDs
2013	Many new column-oriented DBMSs Traditional vendors added features or acquired systems
now	Transition to the Cloud

MonetDB

- Started by Martin Kersten and Peter Boncz at CWI in 1994
- Remove interpreter from database architecture, no expensive code parsing
- Strict columnar architecture and take into account underlying computer architecture
- Only hard-coded operations on simple arrays
- Reduce overhead, improve CPU cache performance
  - less function calls, checkings per column instead of at every single tuple.
  - algorithms faster to work on set of data
- Columns automatically indexed
- Free and open source code under active development

MonetDB – BATs & Relational algebra

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

MonetDB – BATs & Relational algebra

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

```
SELECT id
      ,name
      ,(age - 30) * 50 AS bonus
  FROM people
 WHERE age > 30
```

MonetDB – BATs & Relational algebra

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

```
SELECT id
      ,name
      ,(age - 30) * 50 AS bonus
  FROM people
 WHERE age > 30
```

```
thetasubselect(age,oid,30,">")
```

candidate_list

(void)	(oid)
0	1
1	2
2	5
3	8
4	9

MonetDB – BATs & Relational algebra

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

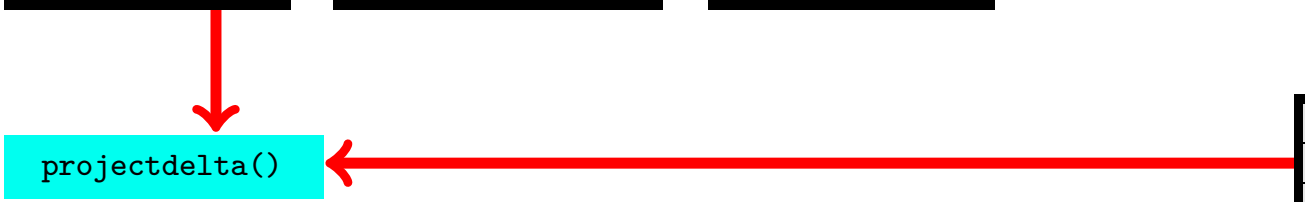
people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

```
SELECT id
      ,name
      ,(age - 30) * 50 AS bonus
  FROM people
 WHERE age > 30
```



candidate_list

(void)	(oid)
0	1
1	2
2	5
3	8
4	9

bat_id

(void)	(int)
0	102
1	104
2	109
3	114
4	115

MonetDB – BATs & Relational algebra

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

```
SELECT id
      ,name
      ,(age - 30) * 50 AS bonus
  FROM people
 WHERE age > 30
```

candidate_list

(void)	(oid)
0	1
1	2
2	5
3	8
4	9

`projectdelta()`

bat_id

(void)	(int)
0	102
1	104
2	109
3	114
4	115

bat_name

(void)	(str)
0	Ivan
1	Peggy
2	Walter
3	Zoe
4	Charlie

MonetDB – BATs & Relational algebra

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

```
SELECT id
      ,name
      ,(age - 30) * 50 AS bonus
  FROM people
 WHERE age > 30
```

candidate_list

(void)	(oid)
0	1
1	2
2	5
3	8
4	9

`projectdelta()`

bat_id

(void)	(int)
0	102
1	104
2	109
3	114
4	115

bat_name

(void)	(str)
0	Ivan
1	Peggy
2	Walter
3	Zoe
4	Charlie

bat_age

(void)	(int)
0	37
1	45
2	31
3	42
4	35

MonetDB – BATs & Relational algebra

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

```
SELECT id
      ,name
      ,(age - 30) * 50 AS bonus
  FROM people
 WHERE age > 30
```

candidate_list

(void)	(oid)
0	1
1	2
2	5
3	8
4	9

bat_id

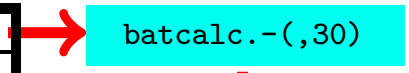
(void)	(int)
0	102
1	104
2	109
3	114
4	115

bat_name

(void)	(str)
0	Ivan
1	Peggy
2	Walter
3	Zoe
4	Charlie

bat_age

(void)	(int)
0	37
1	45
2	31
3	42
4	35



bat_tmp

(void)	(lng)
0	7
1	15
2	1
3	12
4	5

MonetDB – BATs & Relational algebra

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

```
SELECT id
      ,name
      ,(age - 30) * 50 AS bonus
  FROM people
 WHERE age > 30
```

candidate_list

(void)	(oid)
0	1
1	2
2	5
3	8
4	9

bat_id

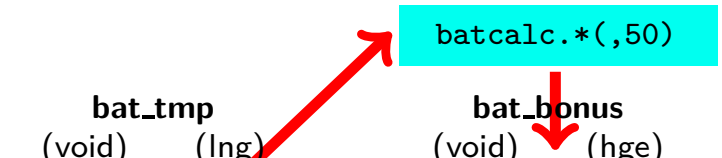
(void)	(int)
0	102
1	104
2	109
3	114
4	115

bat_name

(void)	(str)
0	Ivan
1	Peggy
2	Walter
3	Zoe
4	Charlie

bat_age

(void)	(int)
0	37
1	45
2	31
3	42
4	35



bat_tmp

(void)	(lng)
0	7
1	15
2	1
3	12
4	5

bat_bonus

(void)	(hge)
0	350
1	750
2	50
3	600
4	250

MonetDB – BATs & Relational algebra

people_id

(void)	(int)
0	101
1	102
2	104
3	105
4	108
5	109
6	112
7	113
8	114
9	115

people_name

(void)	(str)
0	Alice
1	Ivan
2	Peggy
3	Victor
4	Eve
5	Walter
6	Trudy
7	Bob
8	Zoe
9	Charlie

people_age

(void)	(int)
0	22
1	37
2	45
3	25
4	19
5	31
6	27
7	29
8	42
9	35

```
SELECT id
      ,name
      ,(age - 30) * 50 AS bonus
  FROM people
 WHERE age > 30
```

candidate_list

(void)	(oid)
0	1
1	2
2	5
3	8
4	9

bat_id

(void)	(int)
0	102
1	104
2	109
3	114
4	115

bat_name

(void)	(str)
0	Ivan
1	Peggy
2	Walter
3	Zoe
4	Charlie

bat_age

(void)	(int)
0	37
1	45
2	31
3	42
4	35

Materialised intermediates

bat_tmp

(void)	(lng)
0	7
1	15
2	1
3	12
4	5

bat_bonus

(void)	(hge)
0	350
1	750
2	50
3	600
4	250

MonetDB – BATs & Relational algebra

```
X_4 := sql.mvc();
X_8:bat[:int] := sql.bind(X_4,"sys","people","age",0);
C_5:bat[:oid] := sql.tid(X_4,"sys","people");
C_78 := algebra.thetasubselect(X_8,C_5,A2,">");
(C_11,r1_11) := sql.bind(X_4,"sys","people","age",2);
C_79 := algebra.thetasubselect(r1_11,nil:bat[:oid],A2,">");
X_14:bat[:int] := sql.bind(X_4,"sys","people","age",1);
C_81 := algebra.thetasubselect(X_14,C_5,A2,">");
C_16 := sql.subdelta(C_78,C_5,C_11,C_79,C_81);
X_18:bat[:int] := sql.bind(X_4,"sys","people","id",0);
(C_20,r1_23) := sql.bind(X_4,"sys","people","id",2);
X_22:bat[:int] := sql.bind(X_4,"sys","people","id",1);
X_23 := sql.projectdelta(C_16,X_18,C_20,r1_23,X_22);
X_24:bat[:str] := sql.bind(X_4,"sys","people","name",0);
(C_26,r1_31) := sql.bind(X_4,"sys","people","name",2);
X_28:bat[:str] := sql.bind(X_4,"sys","people","name",1);
X_29 := sql.projectdelta(C_16,X_24,C_26,r1_31,X_28);
X_30 := sql.projectdelta(C_16,X_8,C_11,r1_11,X_14);
X_31 := batcalc.lng(X_30);
X_32:bat[:lng] := batcalc.-(X_31,A0);
X_33 := batcalc.hge(X_32);
X_34:bat[:hge] := batcalc.*(X_33,A1);
sql.resultSet(X_59,X_61,X_63,X_65,X_67,X_23,X_29,X_34);
```

MonetDB – BATs & Relational algebra

- BAT algebra
  - Hard-coded semantics, operators are predicate-less
  - "RISC approach to database query languages"
- CPU and memory cache optimised
  - CPU/compiler-friendly instructions
- One loop for an entire column
  - no per-tuple interpretation
  - better instruction cache locality

MonetDB – BATs & Relational algebra

- Various search accelerator structures
  - Automatic indexing, built automatically on-the-fly when required/beneficial
  - hash tables, column imprints
- Intermediates can be significant amount of memory, but can be recycled
  - When too large, data is memory-mapped in the system's virtual memory address space.
- Late tuple reconstruction
- Read-optimised

MonetDB – Statistical integration

- Statistical analysis
  - Collect data
  - **Load data**
  - **Filter, transform & aggregate data**
  - Analyse & visualise data
- Bridging the gap between Statistical Toolkits & RDBMS
  - Native operators and lazy evaluation
  - Cheap data transfer
- Integration of statistical toolkits into DBMS
  - Embedded `R`, `MonetDBLite` for `R`
  - Embedded `Python/NumPy`, `MonetDBLite` for `Python` on its way
  - Indexing using *k*-d tree
- UDFs become *really* efficient

MonetDB – Embedded Python

```
CREATE FUNCTION chisq_prob(chisq DOUBLE, dof INT)
RETURNS DOUBLE
LANGUAGE PYTHON {

    from scipy import stats
    return 1 - stats.chi2.cdf(chisq, dof)

};
```

MonetDB – Embedded Python

```
SELECT *
  FROM (SELECT a.runcat
              ,a.xtrsrc
              ,x.image
              ,i.band
              ,a.type
              ,eta_int
              ,f_datapoints
              ,chisq_prob(eta_int * f_datapoints, f_datapoints)
               AS chisqp
          FROM assocxtrsource a
              ,extractedsource x
              ,image i
         WHERE a.xtrsrc = x.id
           AND x.image = i.id
       ) t
 WHERE chisqp < 0.1
   AND f_datapoints > 20
ORDER BY runcat
        ,xtrsrc
```

MonetDB – Embedded Python

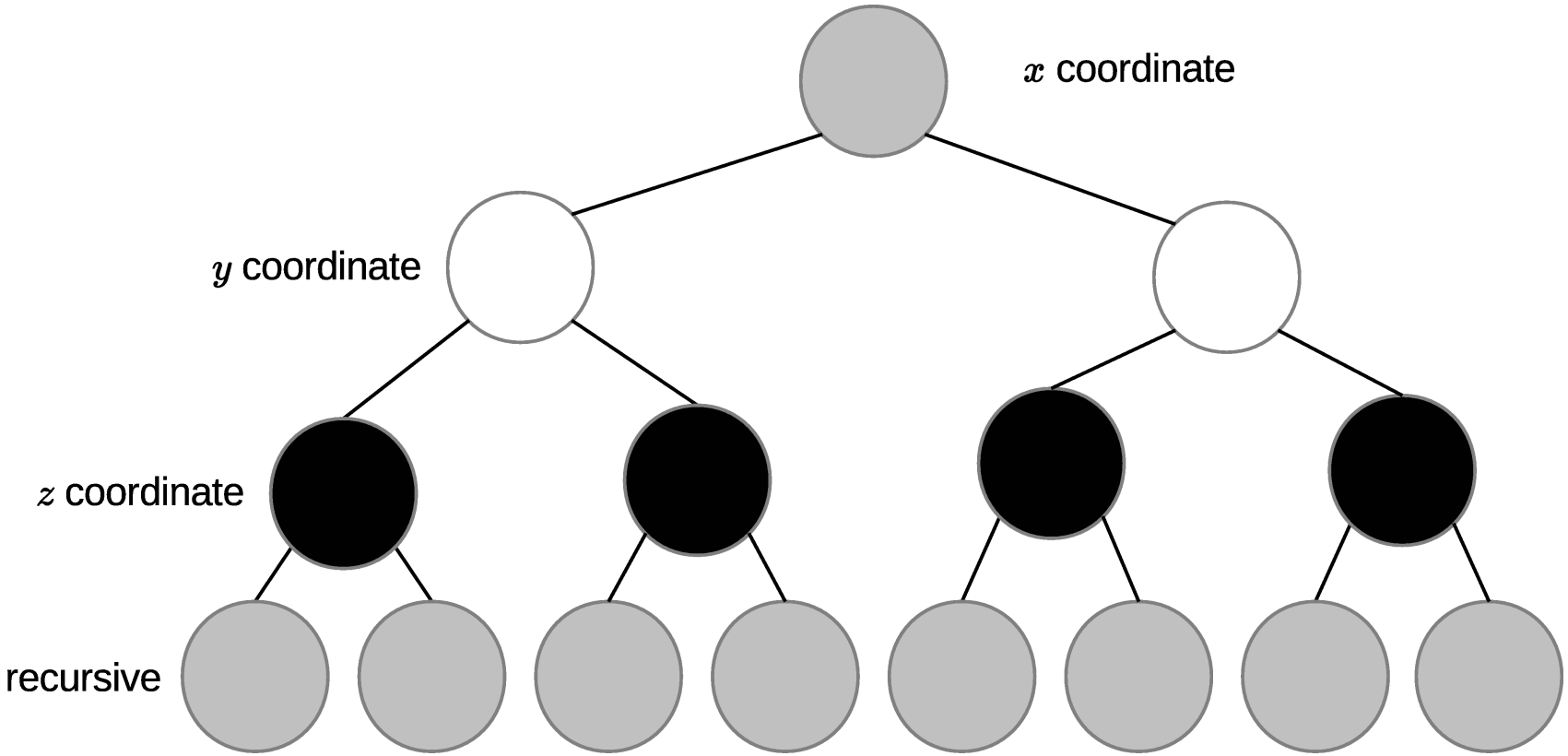
```
SELECT *
  FROM (SELECT a.runcat
              ,a.xtrsrc
              ,x.image
              ,i.band
```

atapoints)

- See ***Vectorized UDFs in Column-Stores*** by M.Raasveldt & H.Mühleisen, SSDBM 2016 for more details
- Mark's blog posts at monetdb.org
- All available in next release

```
          AND x.image = i.id
      ) t
 WHERE chisqp < 0.1
   AND f_datapoints > 20
ORDER BY runcat
        ,xtrsrc
```

MonetDB – k-d tree index – cross-matching



MonetDB – *k*-d tree index – cross-matching

```
SELECT * FROM k3m_free();

SELECT * FROM k3m_build((SELECT id
                              ,ra
                              ,decl
                          FROM catalog
                        ));
```

MonetDB – k-d tree index – cross-matching

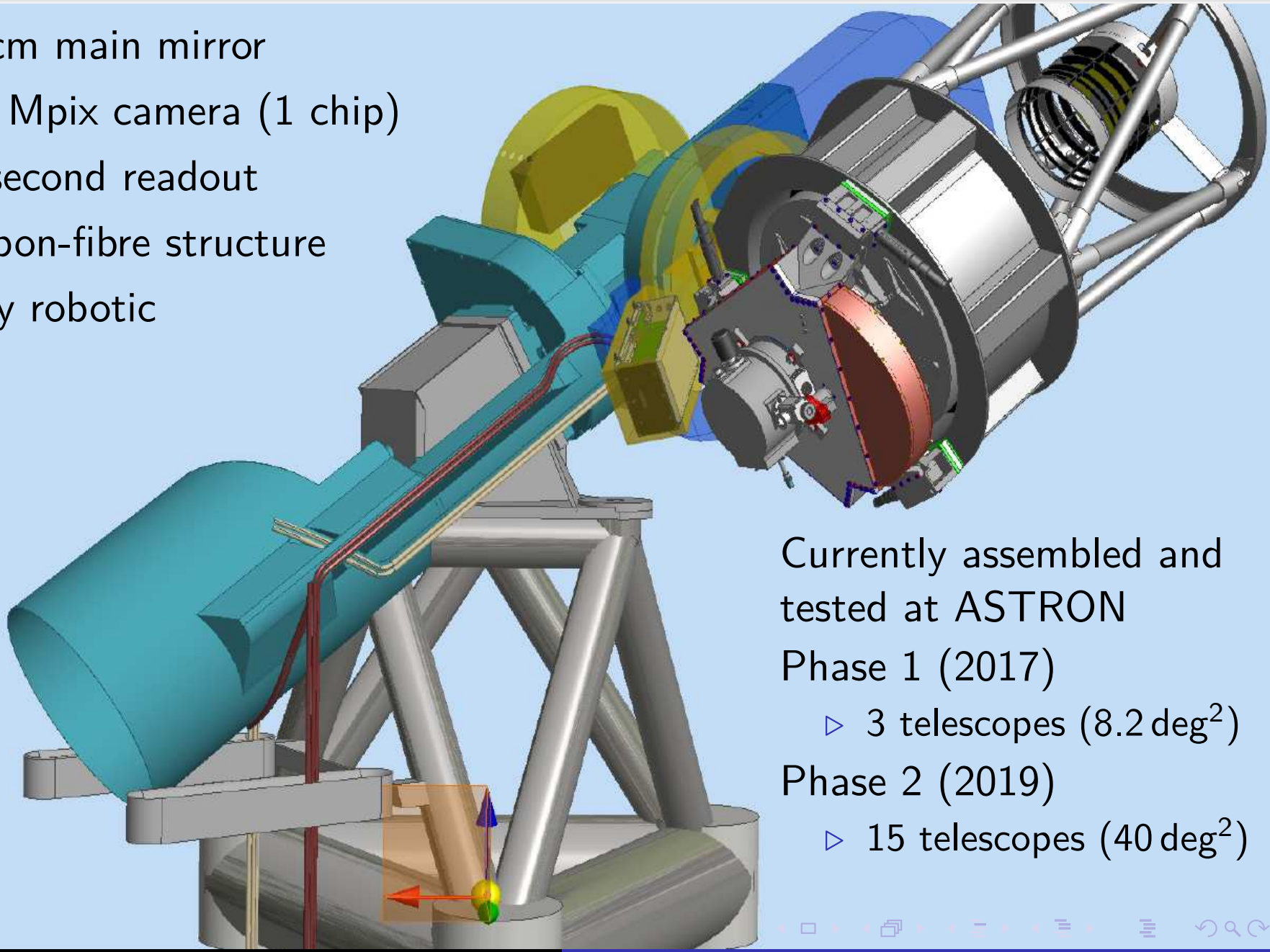
```
DECLARE search_radius DOUBLE;
SET search_radius = 0.5;  /* some unit */

SELECT idc
      ,ids
      ,dist_rad
  FROM k3m_query((SELECT id
                       ,ra
                       ,decl
                       ,search_radius
                   FROM extractedsource
                ))
;
```

Context of high-cadence astronomy

- Time is a new dimension
- Keep all measurements/source properties
  $\Rightarrow$ constructing light curves, spectra, color–color plots
- Actively use database
  $\Rightarrow$ move algorithms and statistics inside database engine
  - Maintain statistical sky model
  - Cross-match
  - Process in bulk mode
- Pipeline requires quick responses
  $\Rightarrow$ single node
- Data accumulate over time
  $\Rightarrow$ multiple nodes

BlackGEM – The optical telescope

- 65 cm main mirror
- 110 Mpix camera (1 chip)
- 10 second readout
- Carbon-fibre structure
- Fully robotic



Currently assembled and tested at ASTRON

Phase 1 (2017)

- 3 telescopes (8.2 $\deg^2$)

Phase 2 (2019)

- 15 telescopes (40 $\deg^2$)

MeerLICHT – The multi-wavelength telescope

- Single telescope of BlackGEM type in South Africa
- "Attached" to MeerKAT radio telescope
- Pointing determined by MeerKAT
- Changing transient science to truly multi-wavelength
- In South Africa: bridge between SALT and SKA/MeerKAT

BlackGEM/MeerLICHT

Telescope mirror		65 cm	
FoV		$2.7\,\mathrm{deg}^2$	
Resolution		$0.57''/\mathrm{px}$	
CCD size		$10{,}536 \times 10{,}536 = 1.11 \times 10^8$	
bits per px		16 b	
Image size		1.78 Gb = 222 MB	
Cal. images per night		$2 \times (10\,\mathrm{bias} + 5 \times 5\,\mathrm{flats}) = 70$	
DB source data size		402 B	
Observation mode		nominal	**fast**
Integration time		5 min	1 min
Sensitivity		23 mag	21 mag
Peak source density			$300{,}000\,\mathrm{deg}^{-2}$
Av. source density			$35{,}000\,\mathrm{deg}^{-2}$
Science images per night		120	600
Data rate per night		337 Gb = 42 GB	1.2 Tb = 148 GB
Transfer rate	real time	6 Mb/s	30 Mb/s
	24hrs	3.9 Mb/s	13.8 Mb/s
	10hrs	9.4 Mb/s	33.1 Mb/s

BlackGEM/MeerLICHT

Telescope mirror		65 cm	
FoV		$2.7\,\mathrm{deg}^2$	
Resolution		$0.57''/\mathrm{px}$	
CCD size		$10{,}536 \times 10{,}536 = 1.11 \times 10^8$	
bits per px		16 b	
Image size		1.78 Gb = 222 MB	
Cal. images per night		$2 \times (10\,\mathrm{bias} + 5 \times 5\,\mathrm{flats}) = 70$	
DB source data size		402 B	
Observation mode		nominal	**fast**
Integration time		5 min	1 min
Sensitivity		23 mag	21 mag
Peak source density			$300{,}000\,\mathrm{deg}^{-2}$
Av. source density			$35{,}000\,\mathrm{deg}^{-2}$
Science images per night		120	600
Data rate per night		337 Gb = 42 GB	1.2 Tb = 148 GB
Transfer rate	real time	6 Mb/s	30 Mb/s
	24hrs	3.9 Mb/s	13.8 Mb/s
	10hrs	9.4 Mb/s	33.1 Mb/s

SciLens Platform – Experimentation cluster

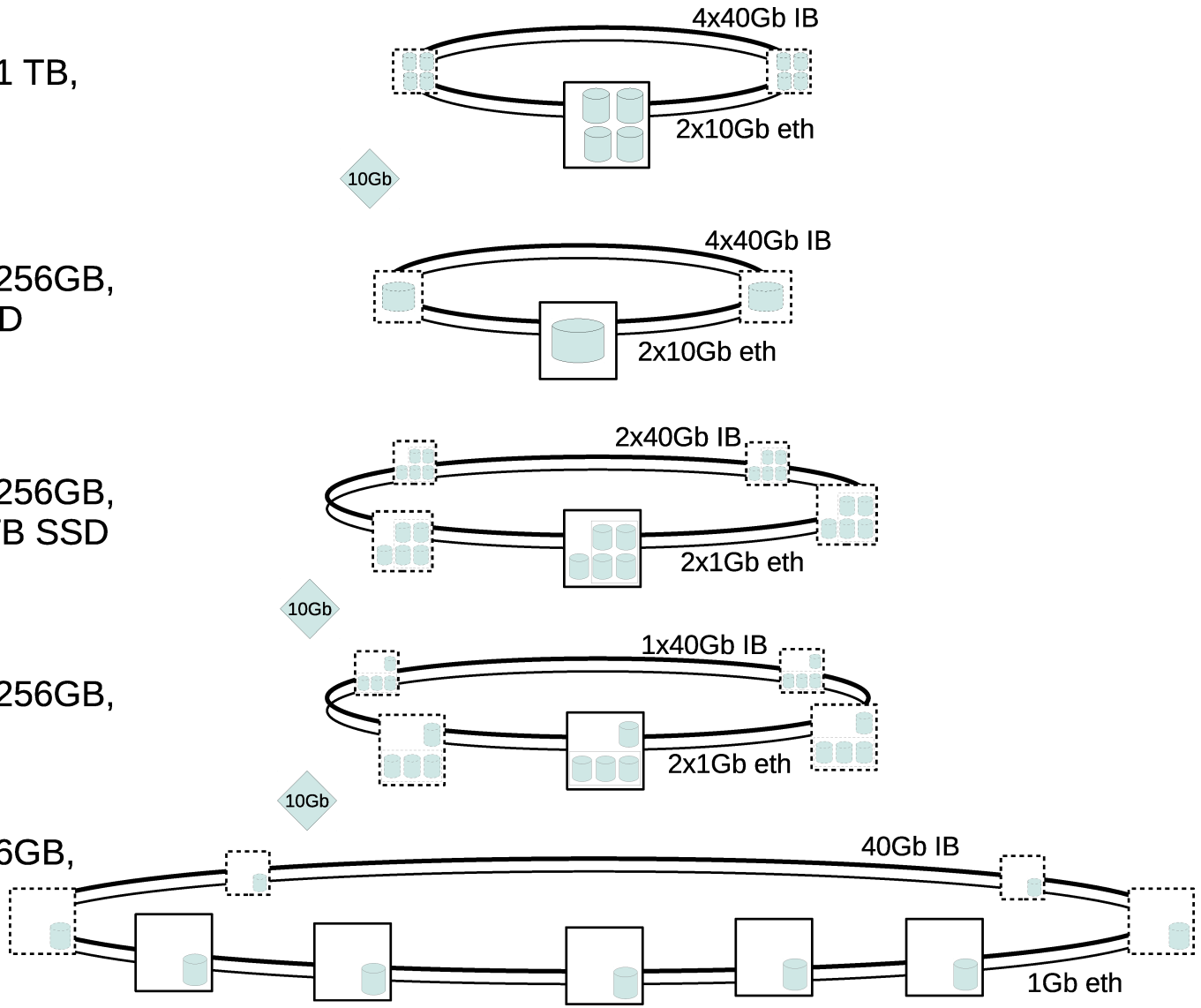
4 **diamonds**
Intel 96 core 1 TB,
4x2TB HDD

4 **gems**
Intel 32 core 256GB,
24x0.5TB SSD

16 **bricks**
Intel 32 core 256GB,
8TB HDD, 1TB SSD

16 **stones**
Intel 32 core 256GB,
12TB HDD

32 **rocks**
Intel 8 core 16GB,
2TB HDD



SciLens Platform — Cluster configuration

		diamonds	stones	bricks	rocks
CPU	Architecture	x86_64			
	CPU(s)	96	32		8
	Threads per core	2			
	Cores per socket	12	8		4
	Socket(s)	4	2		1
	Clockspeed	2.4 – 2.9 GHz	2.6 – 3.4 GHz	2.0 – 2.8 GHz	3.4 – 3.8 GHz
RAM	Size	1024 GB	256 GB		16 GB
SSB	Drives			8 × 128 GB	
	/ssd			8x HW RAID0	
HDD	Disks	4 × 2 TB	3 × 3 TB	4 × 2 TB	1 × 2 TB
	/scratch	7.2 TB (4× HW RAID0)	5.4 TB (3× SW RAID0)		1.8 TB
	/data		3×0.9 TB	1.8 TB	
Network	Ethernet	2 × 10 Gb/s	1 Gb/s		
	Infiniband	4 × 40 Gb/s	40 Gb/s		
Software	OS	Linux, Fedora 24 4.7.3-200.fc24			
	MonetDB	Jun2016 SP1			
	Python	2.7.12			

https://www.monetdb.org/wiki/Scilens-configuration-standard

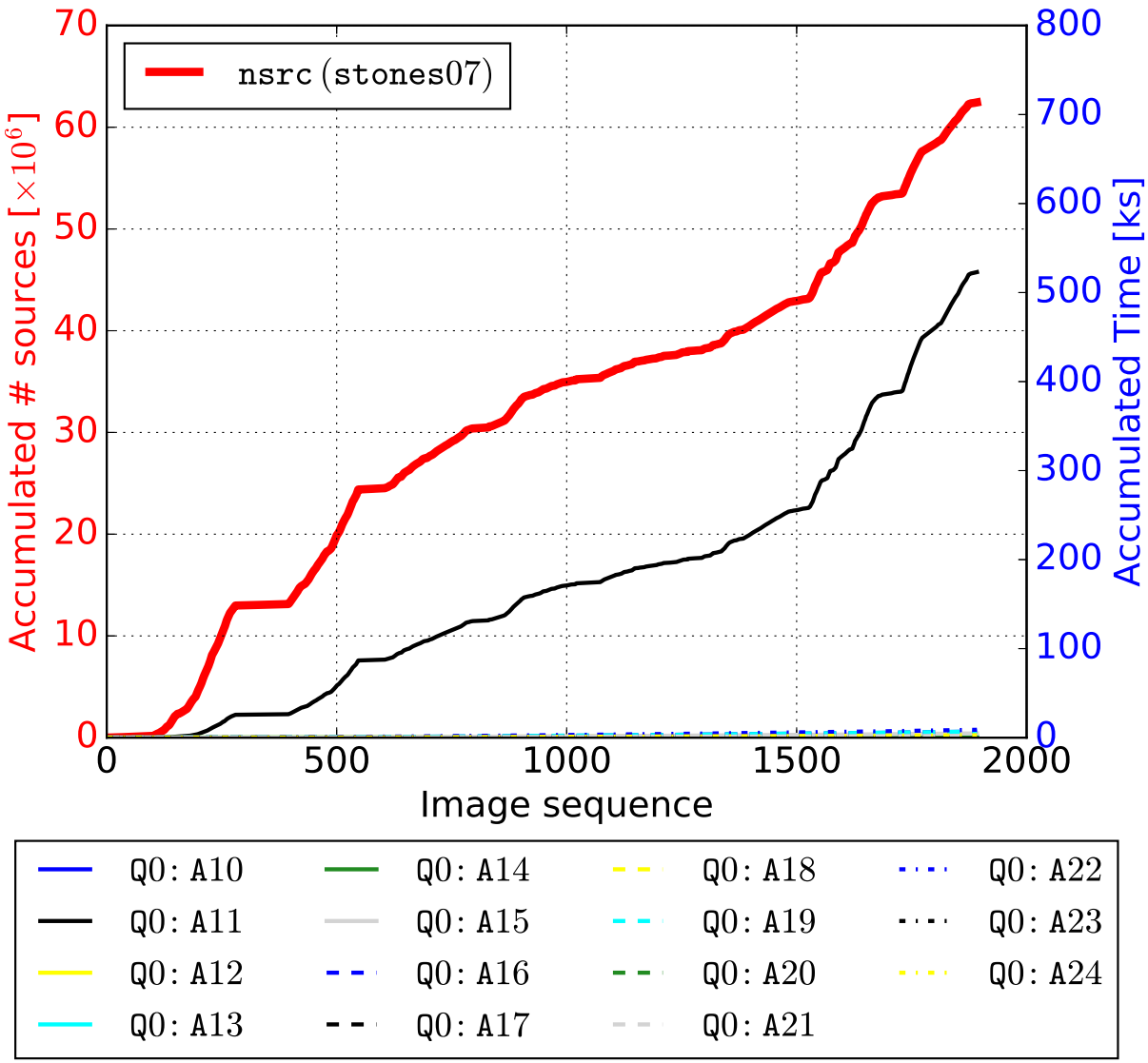
Performance tests with IPHAS data

- IPHAS source densities ∼comparable to BlackGEM
- IPHAS, mapping northern Galactic Plane
- magnitude 20 (10σ)
- FoV: $0.29\,\mathrm{deg}^2$
- about 200 observations per night
- single observation has several 1000's up to 150,000 sources

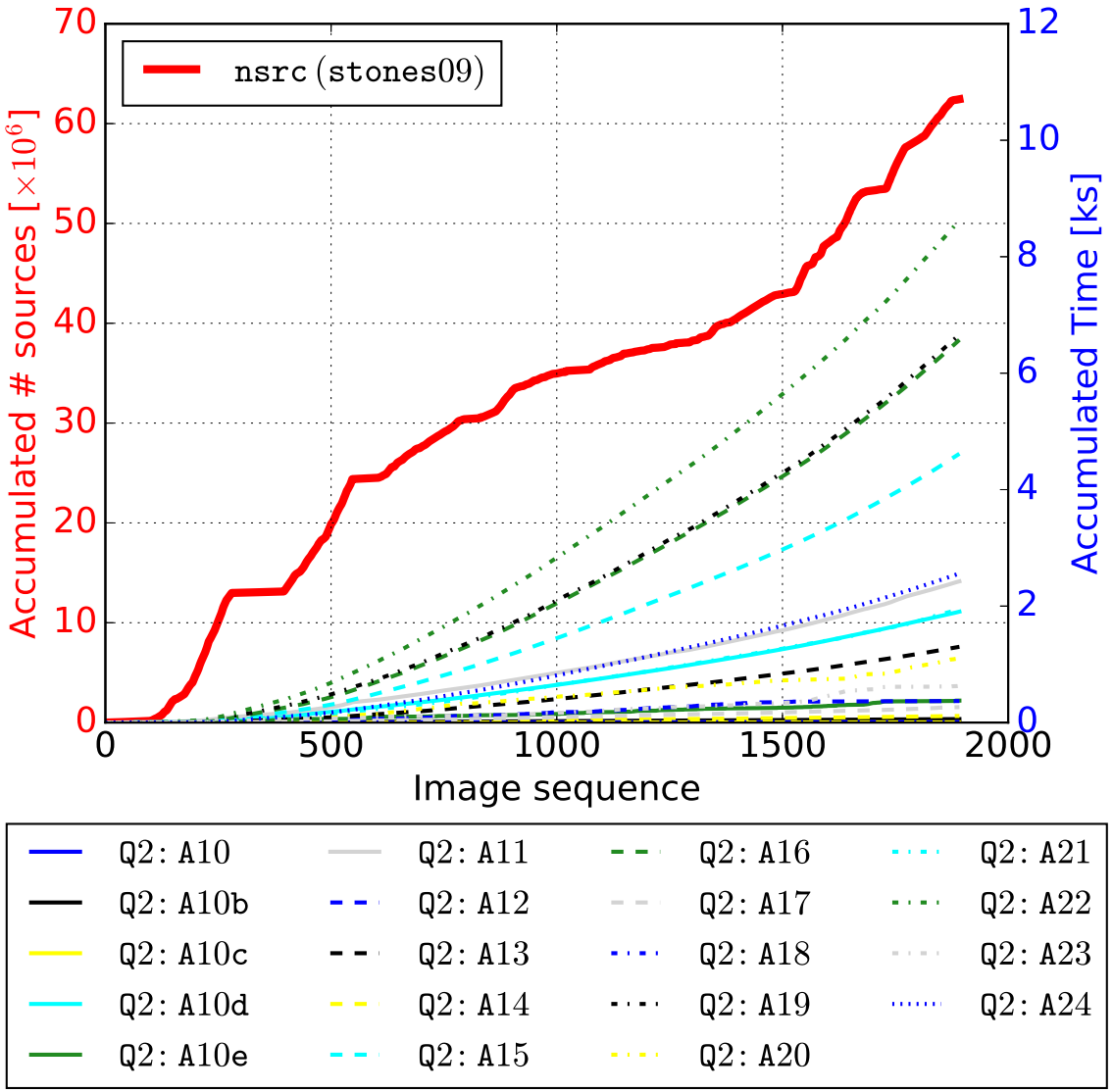
Performance tests with IPHAS data

- Use May 2007 data, 1893 binary `FITS` files
- Monitor/benchmark all queries
  - Insert all sources measured
    $\Rightarrow$ Support of SQL/MED (Management of External Data)
  - Cross-match batch of new sources with counterparts in catalog
  - Update catalog, build flux time series (light curves)
- Pipeline database grows and changes over time:
  dynamic database $\Rightarrow$ indexes get unbalanced
- Try several Query modules (40–50 queries)
  - Q0: baseline, adopted from LOFAR's `TraP`
  - Q2: Use temporary tables sorted by declination
  - Q5: Q2 + (Horizontal) partitioning of data by declination zones of 1-degree width

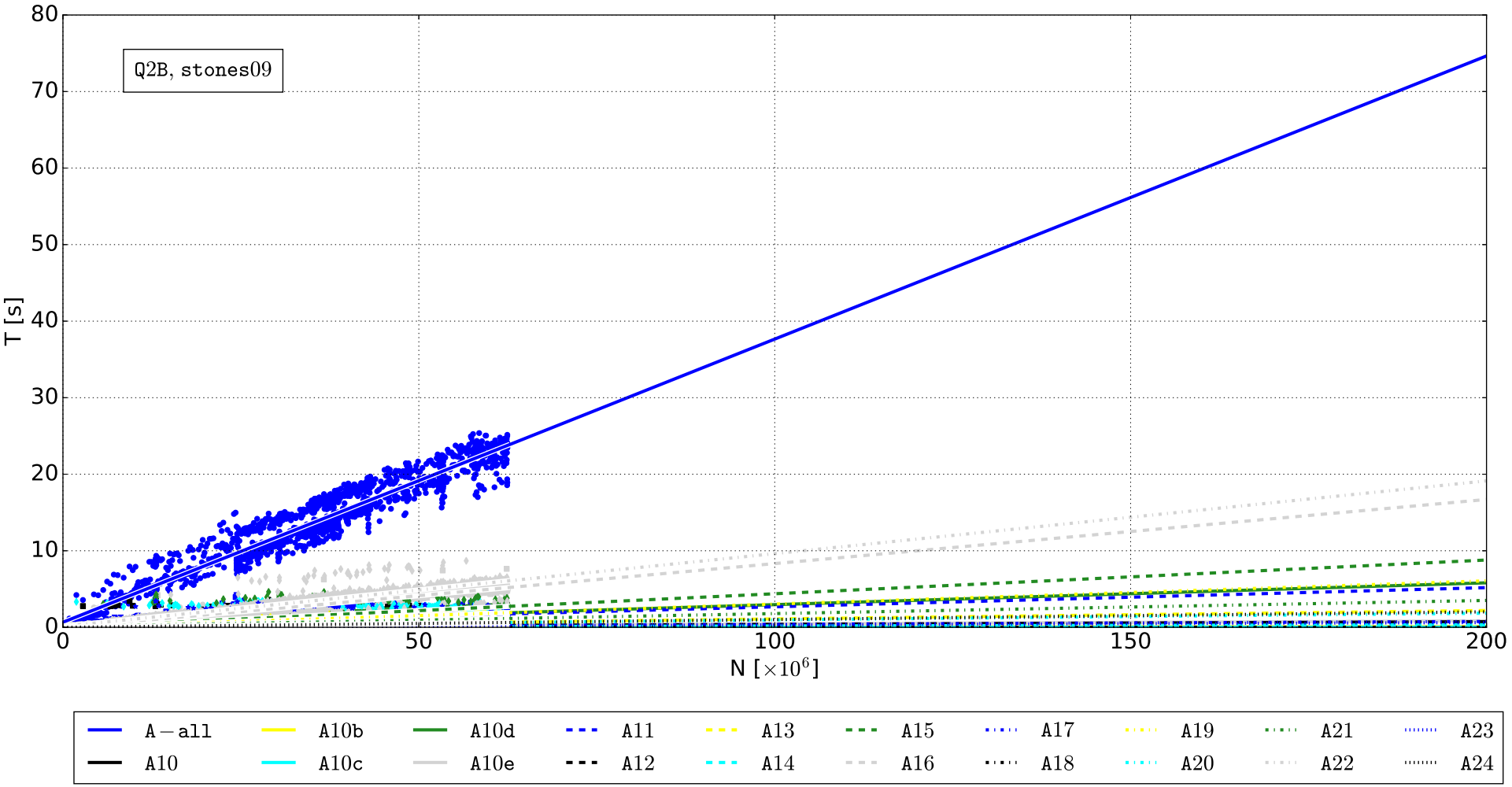
Q0 – Baseline


nsrc (stones07)
Accumulated # sources [$\times 10^6$]
Accumulated Time [ks]
Image sequence
0
10
20
30
40
50
60
70
0
100
200
300
400
500
600
700
800
0
500
1000
1500
2000
Q0: A10
Q0: A11
Q0: A12
Q0: A13
Q0: A14
Q0: A15
Q0: A16
Q0: A17
Q0: A18
Q0: A19
Q0: A20
Q0: A21
Q0: A22
Q0: A23
Q0: A24

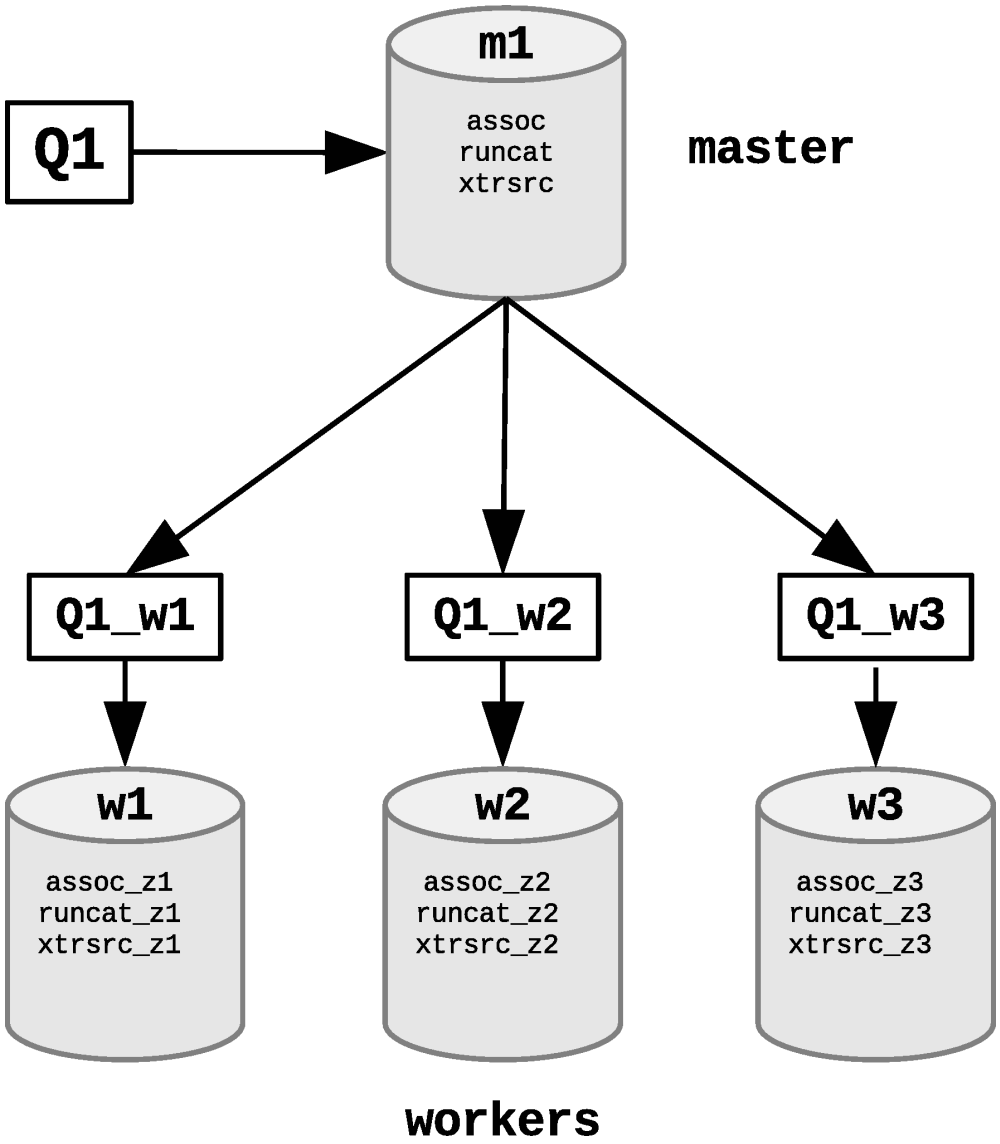
Q2 – Temporary table sorted by declination



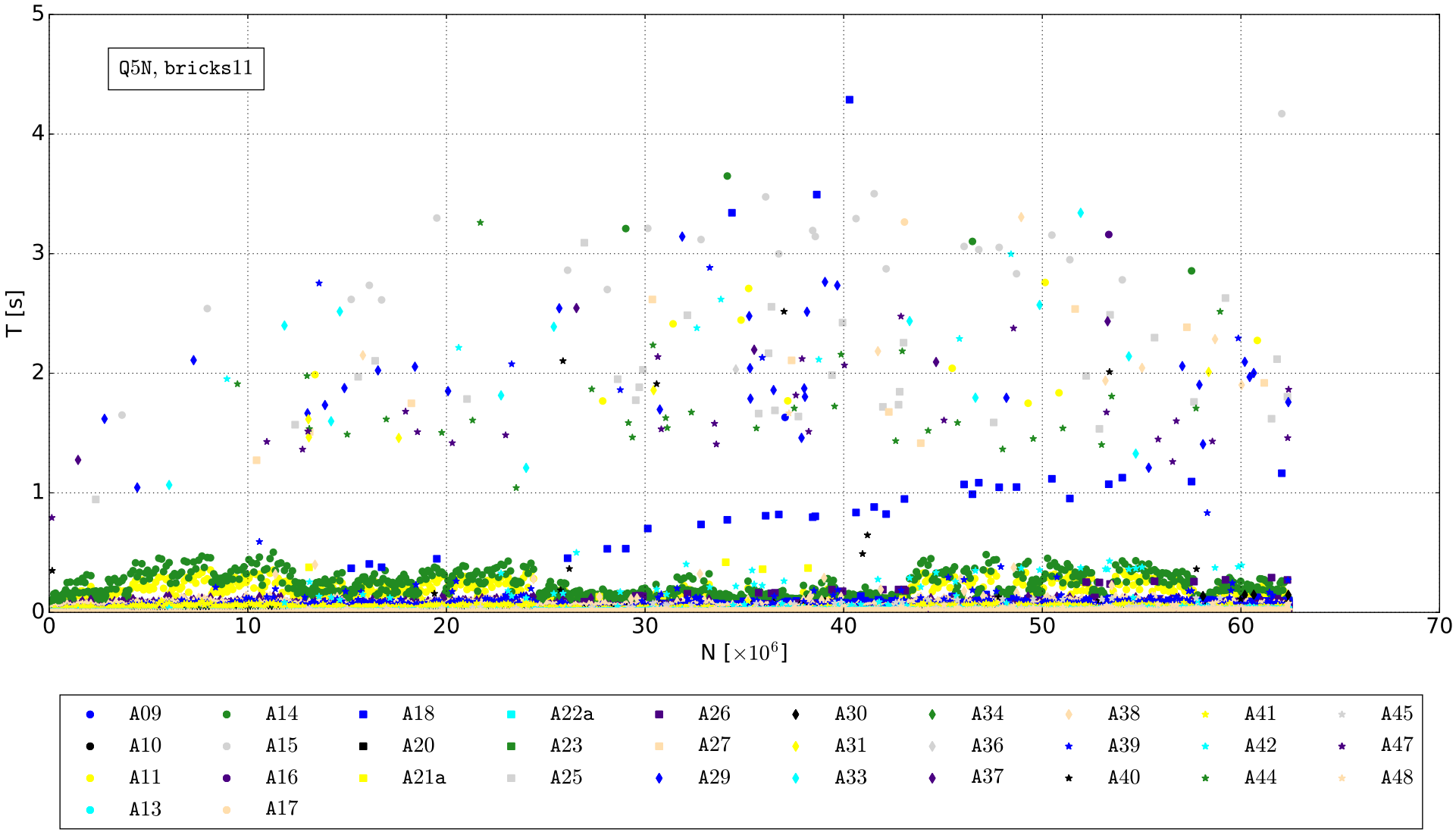
Q2 scales $\sim \mathcal{O}(N)$



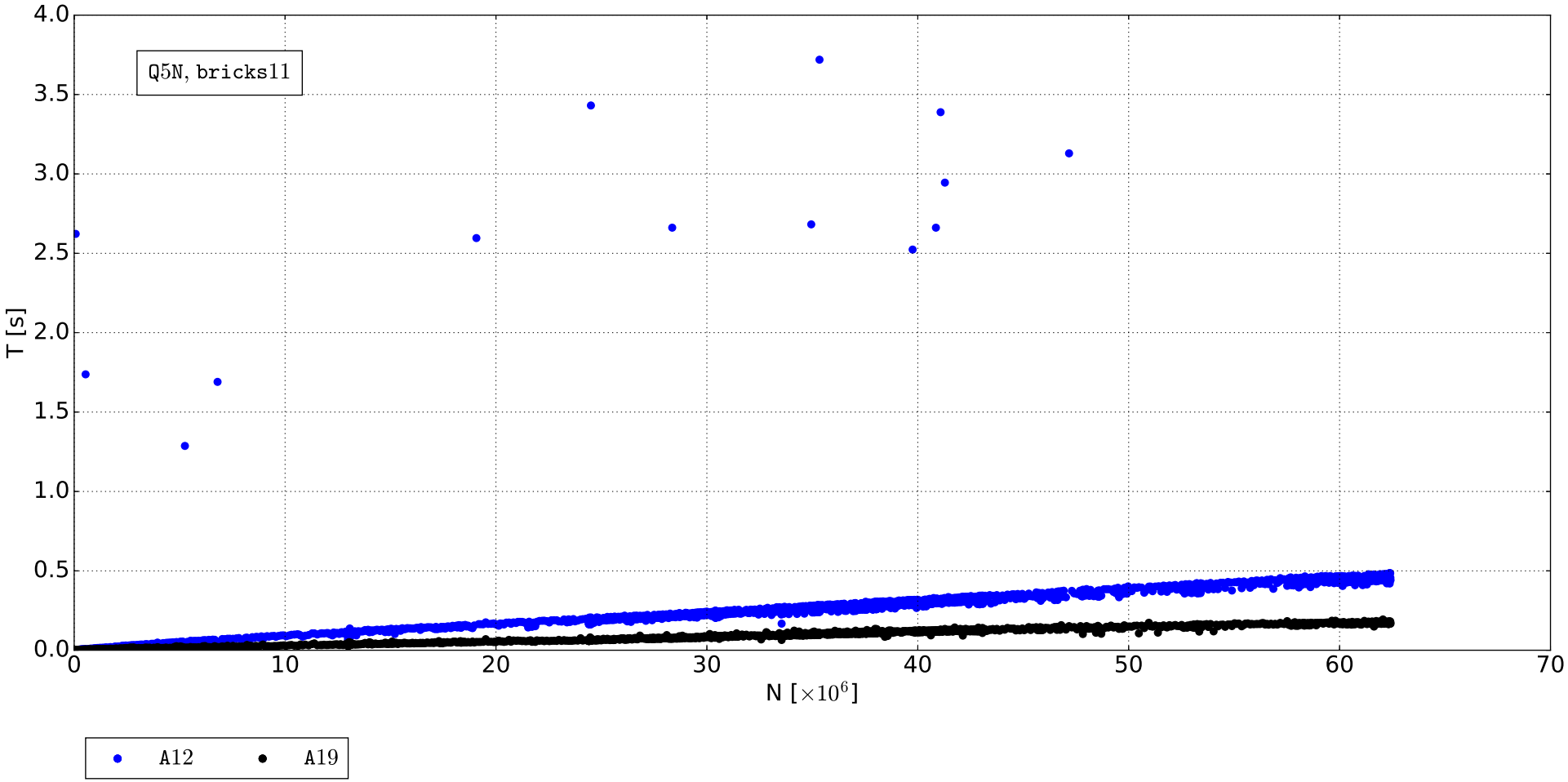
Distributed Querying – Partitioning data



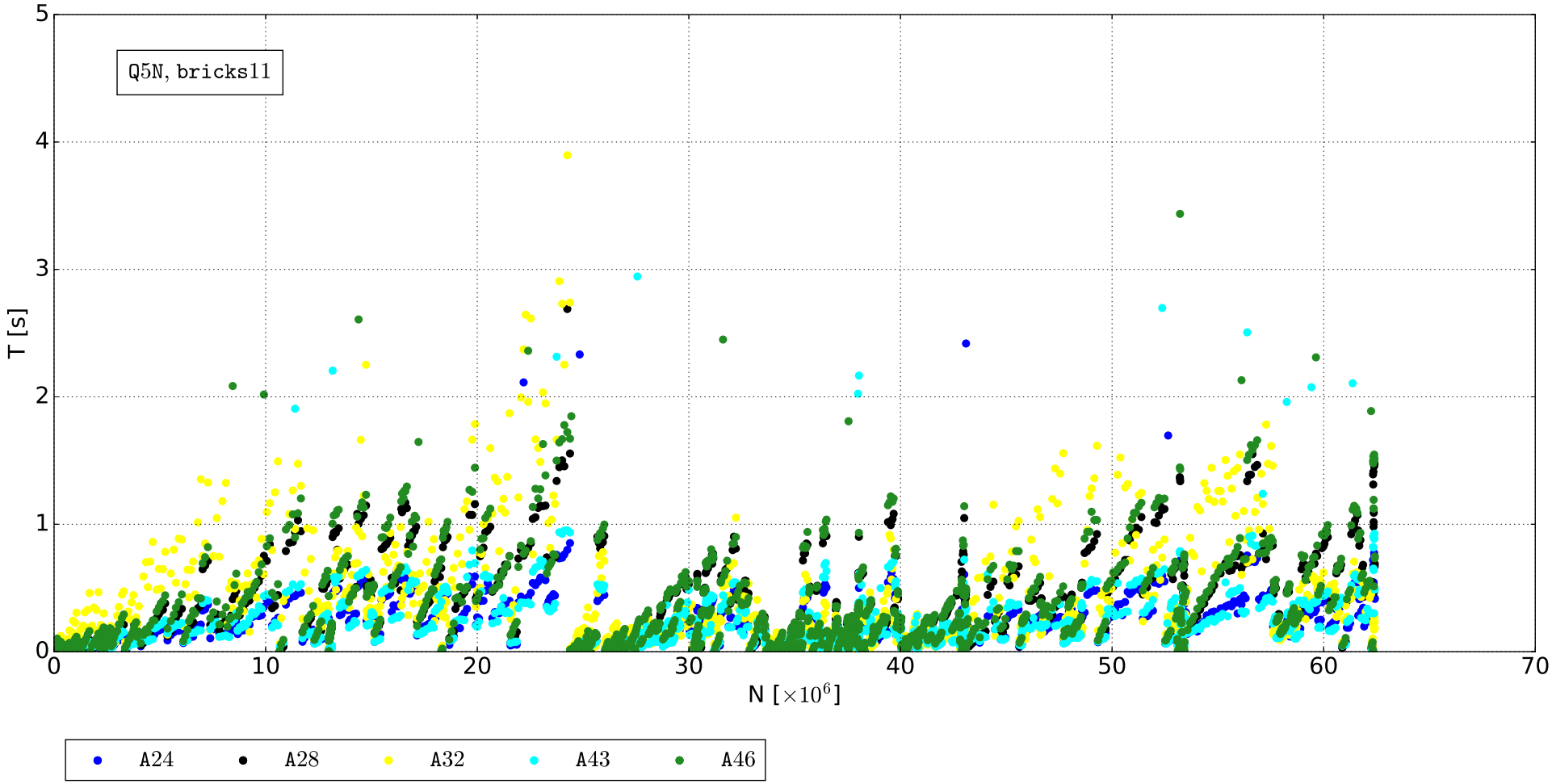
Q5 – Most queries $\sim \mathcal{O}(1)$



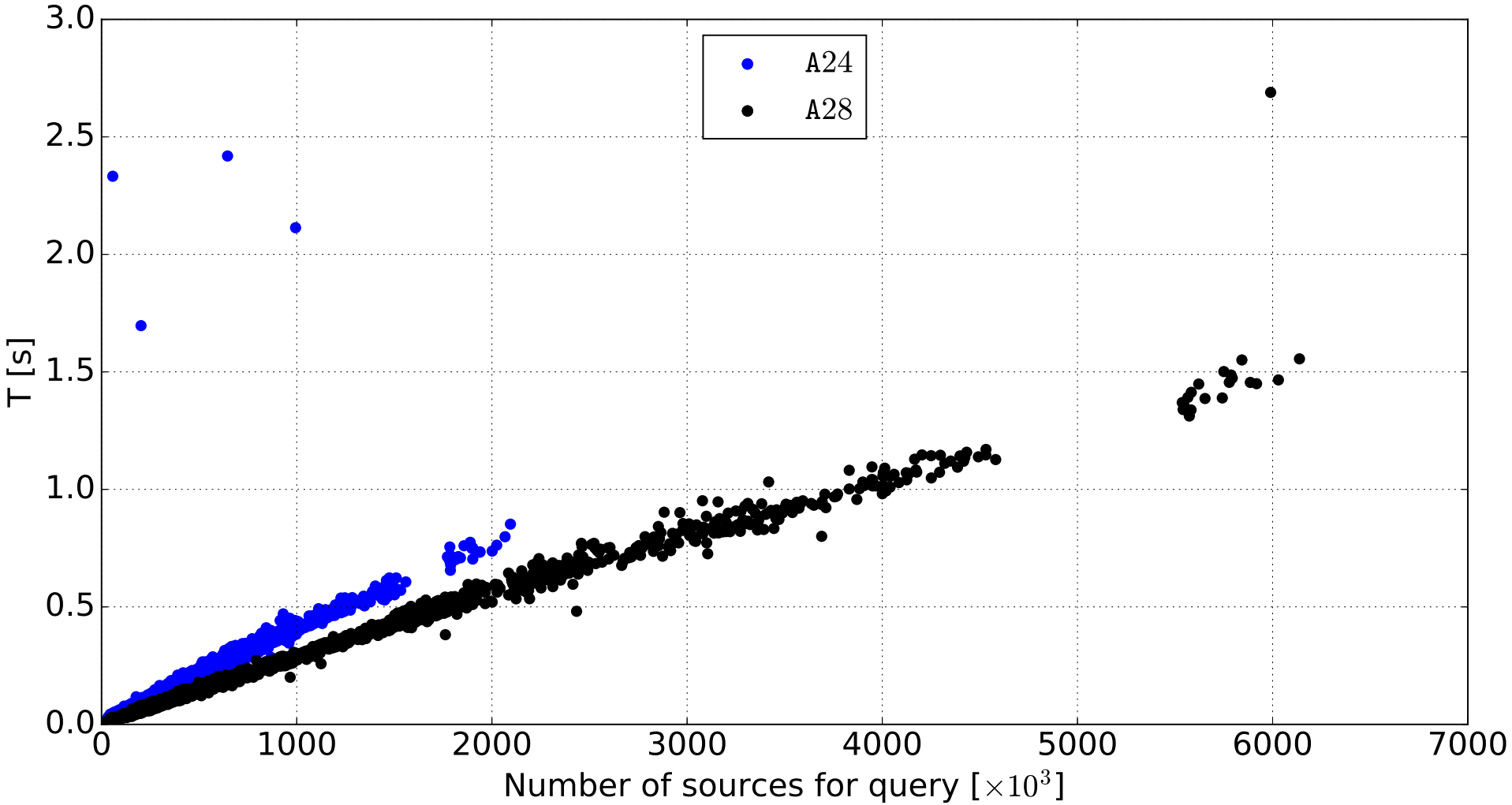
Q5 – SELECT max(id) $\sim \mathcal{O}(N)$



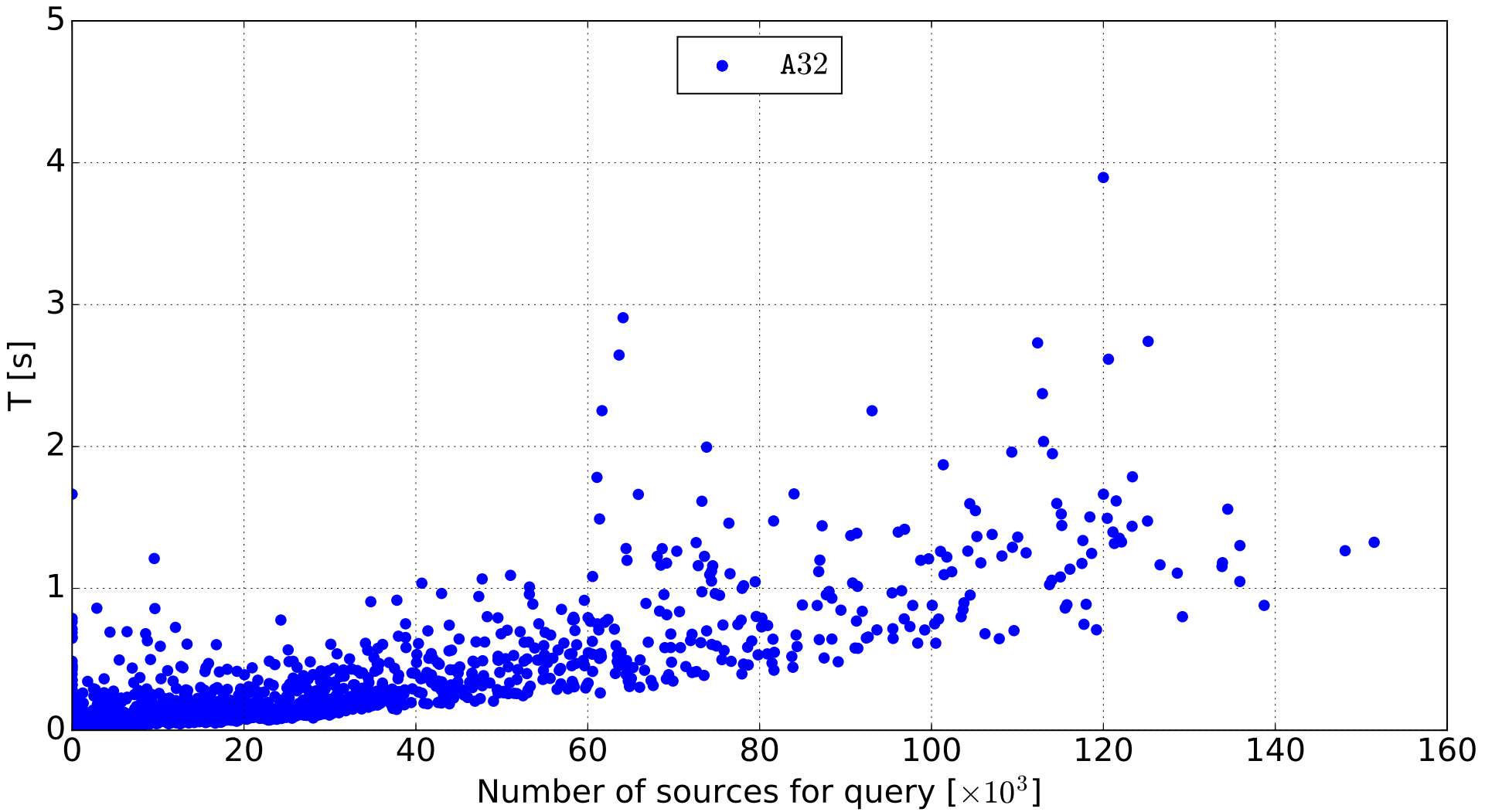
Q5 – Loading $\sim \mathcal{O}(N)$, but database size independent



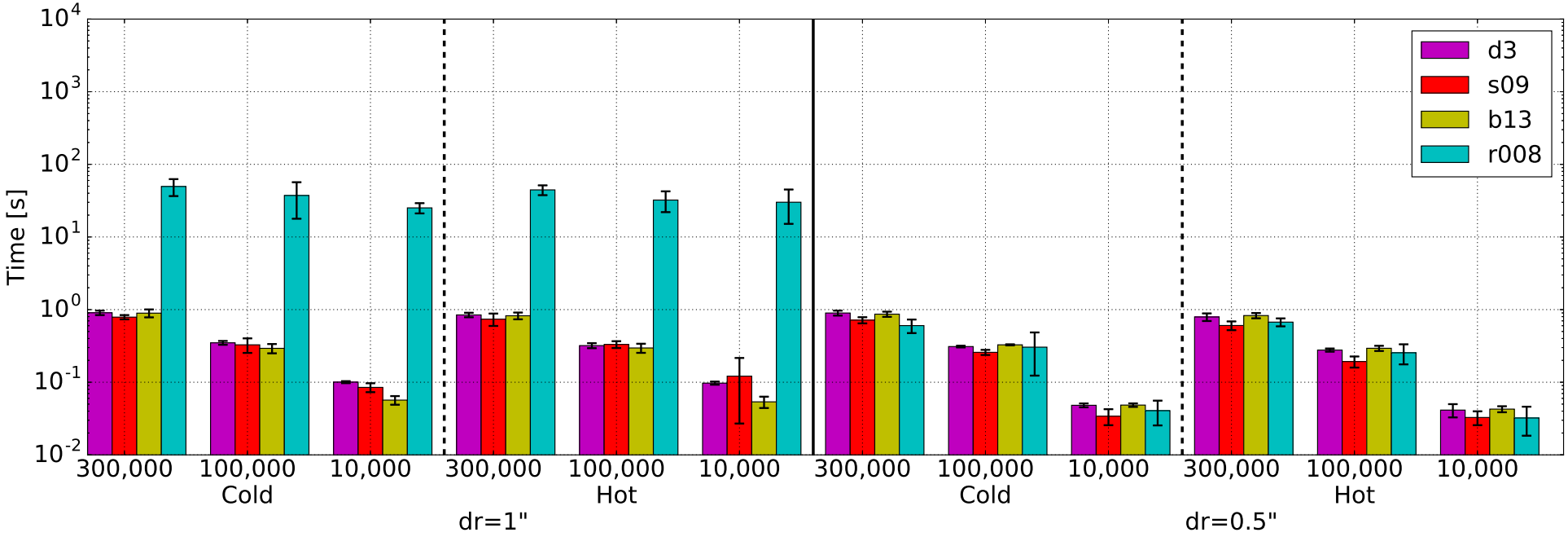
Q5 – Loading $\sim \mathcal{O}(N)$, but database size independent



Q5 – Cross-matching $\sim \mathcal{O}(\log N)$



Q3 – Cross-matching w/ k-d tree index on static database



Conclusions/Outlook

- Conclusions
  - Column-stores fit to high-cadence astronomy data challenges
  - Long-term query monitoring $\Rightarrow$ benchmarking
  - Maintain up-to-date statistical sky model
  - Sorted temporary tables and 3-d binary tree improve cross-matching (latter especially in *static* database)
  - Partition data in dynamic (growing) databases. This reduces query run-times significantly. Opens up processing of SKA and LSST source densities
- Applies to other fields as well
- Outlook
  - Enhance distributed querying to remote/layered/Cloud(Open Stack) cluster nodes
  - Validate database (schema) for *offline* querying
  - Data visualisation

More info on MonetDB & Acknowledgements

- `www.monetdb.org/blog`
  - Data vaults, attach and load `FITS` files
  - Time-domain radio astronomy with MonetDB
  - Embedded `Python/NumPy` and `R` in MonetDB
  - MonetDBLite for `R`
- H2020-Astronomy ESFRI and Research Infrastructure Cluster (Grant Agreement number: 653477).