Runtime Fragility in Main Memory

Endre Palatinus*, Jens Dittrich
Saarland University, Germany

Robust Query Processing

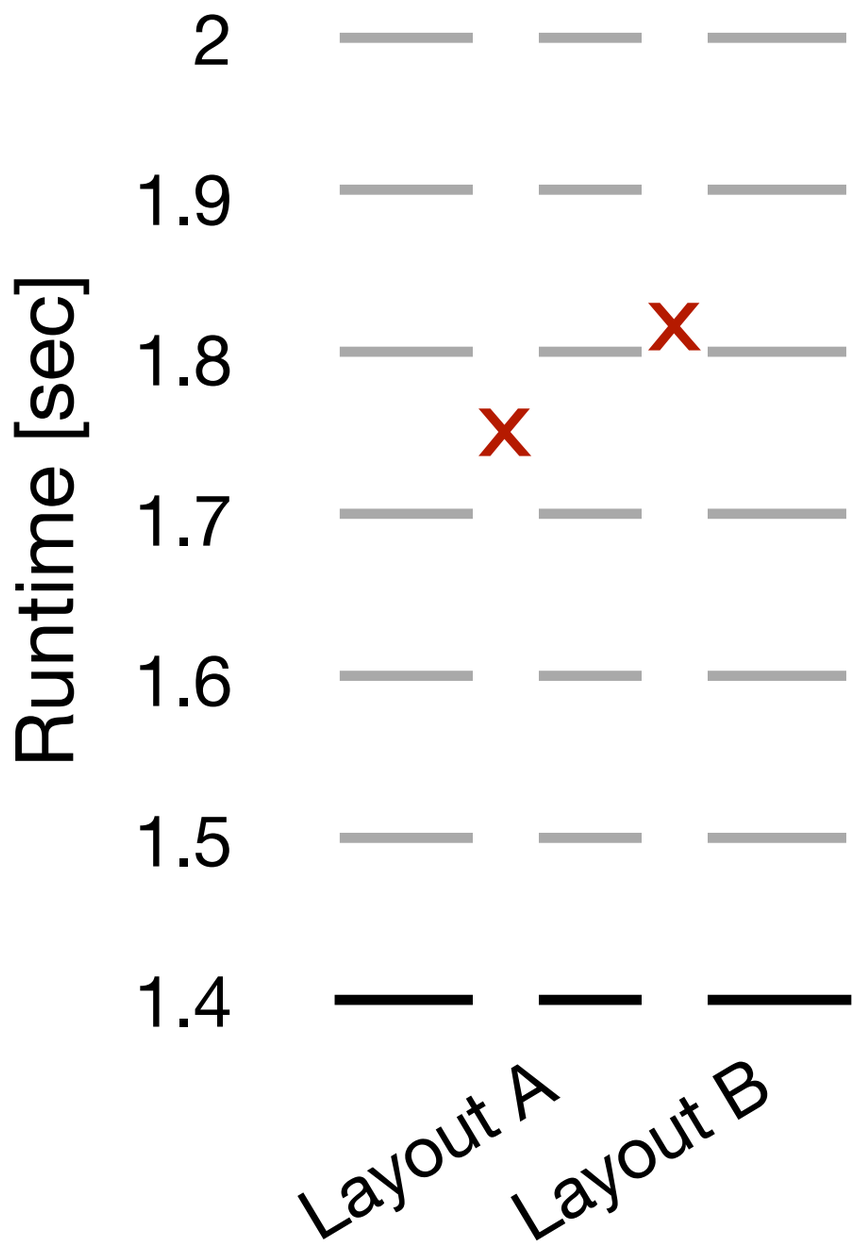
Factors endangering...

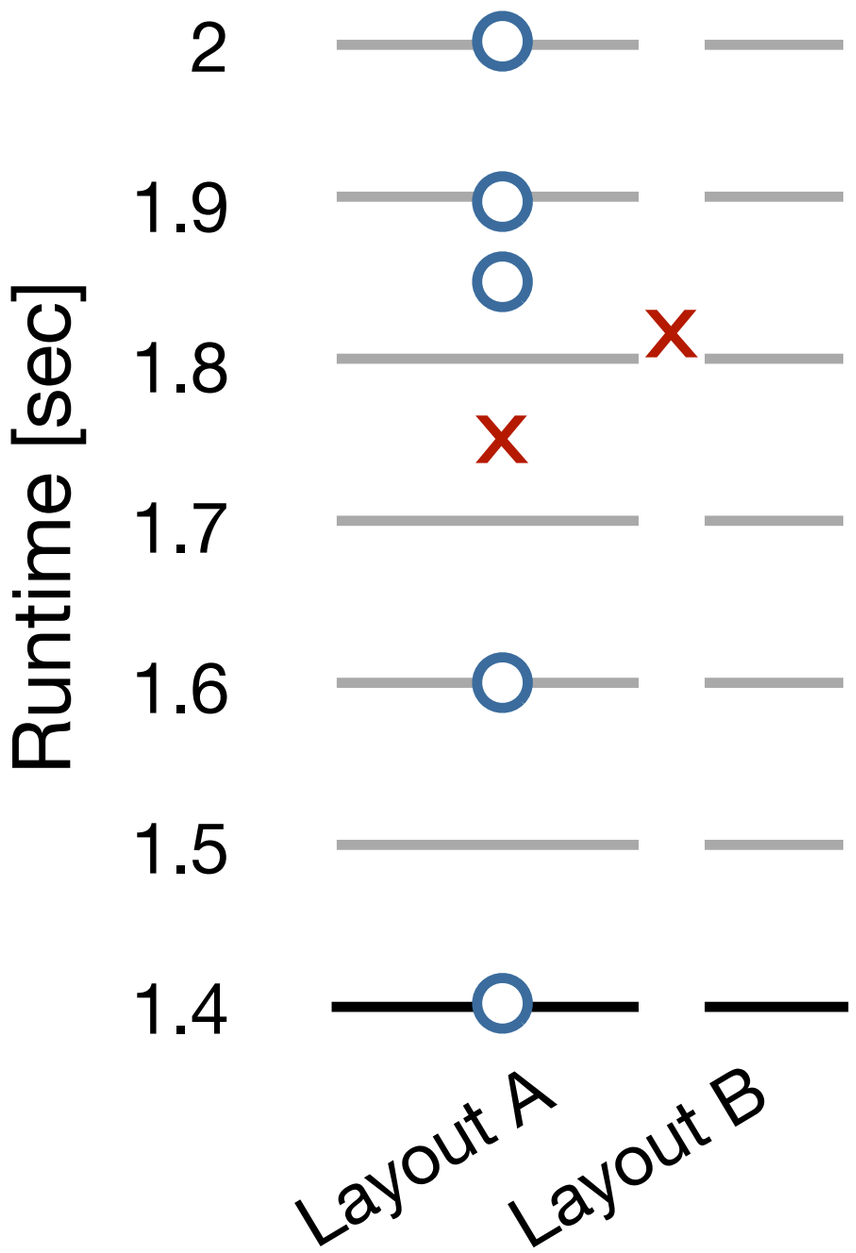
Robust Query Processing

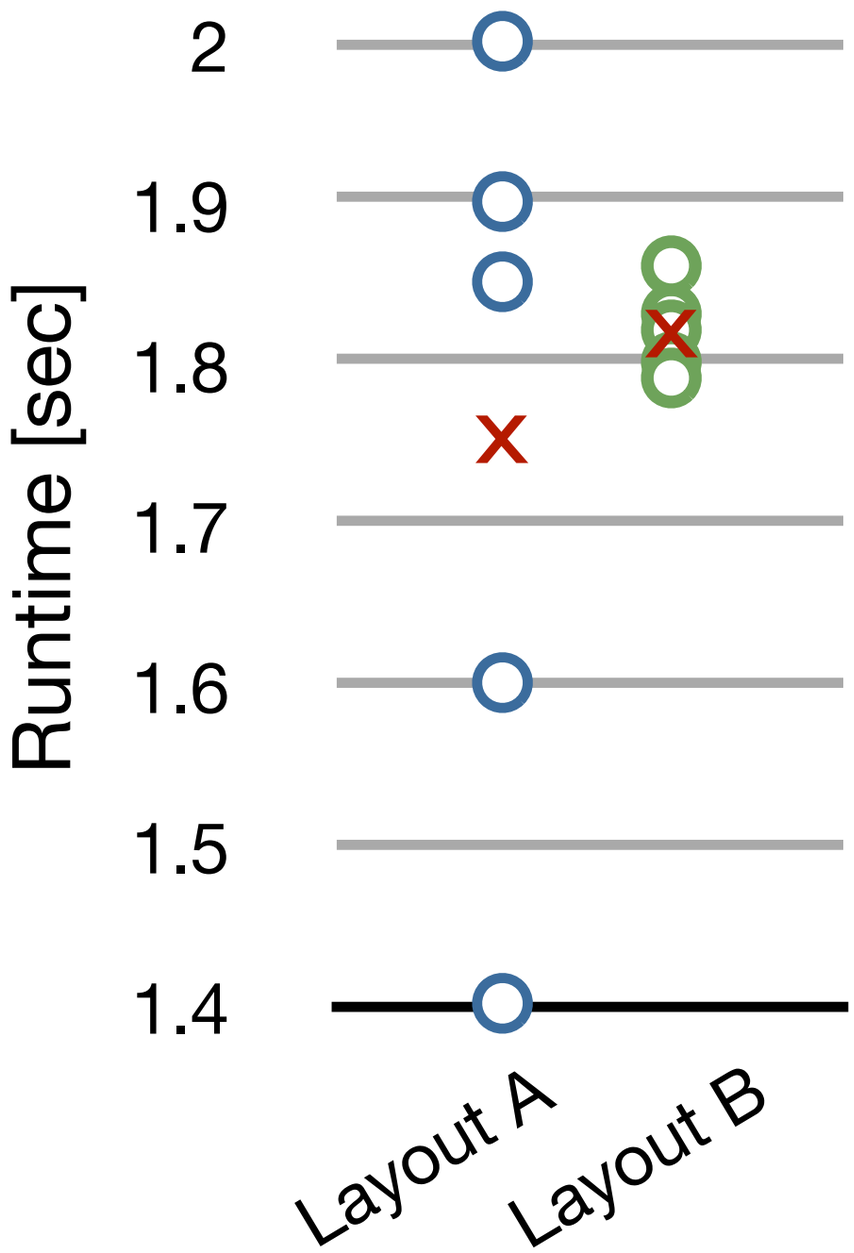
Factors endangering...

Robust Query Processing

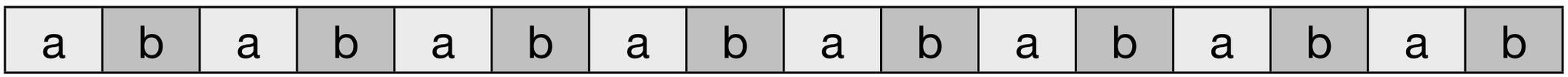
- Wrong cardinality estimates (Victor Leis)
- Predictable execution times (IMDB tutorial)


Runtime [sec]
2
1.9
1.8
1.7
1.6
1.5
1.4
Layout A
Layout B


Runtime [sec]
2
1.9
1.8
1.7
1.6
1.5
1.4
Layout A
Layout B


Runtime [sec]
2
1.9
1.8
1.7
1.6
1.5
1.4
Layout A
Layout B

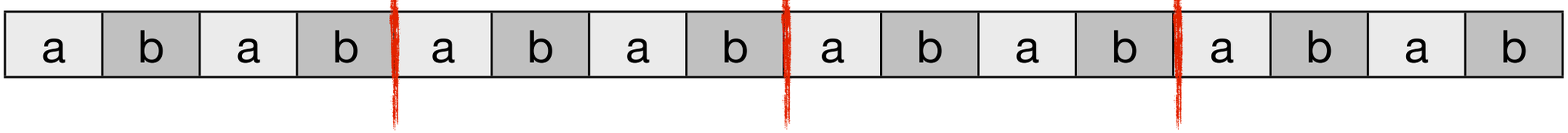
Row



Column

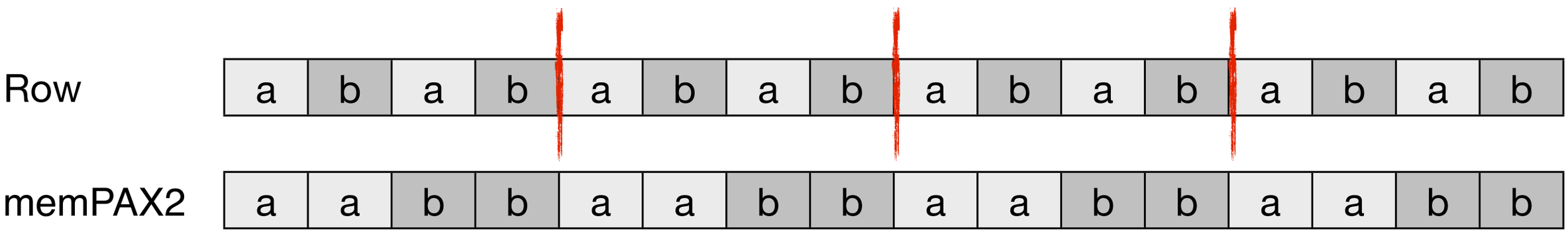


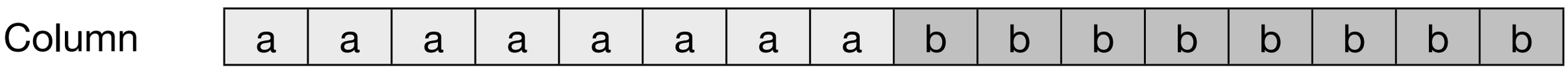
Row

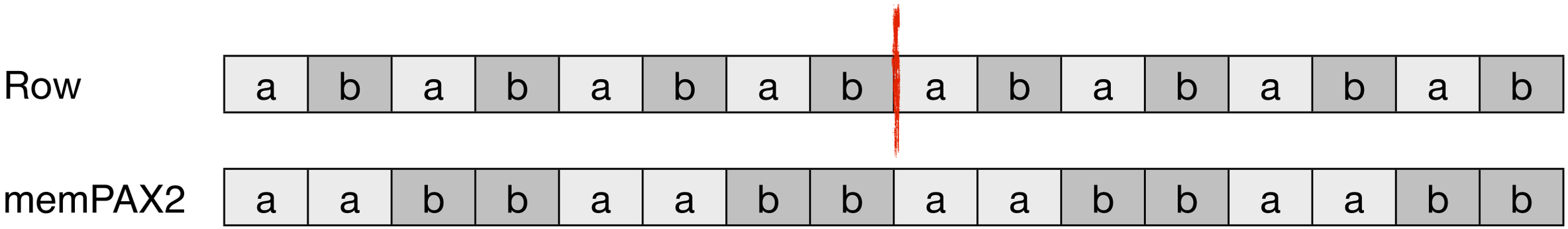


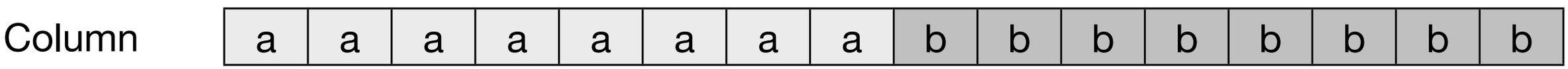
Column

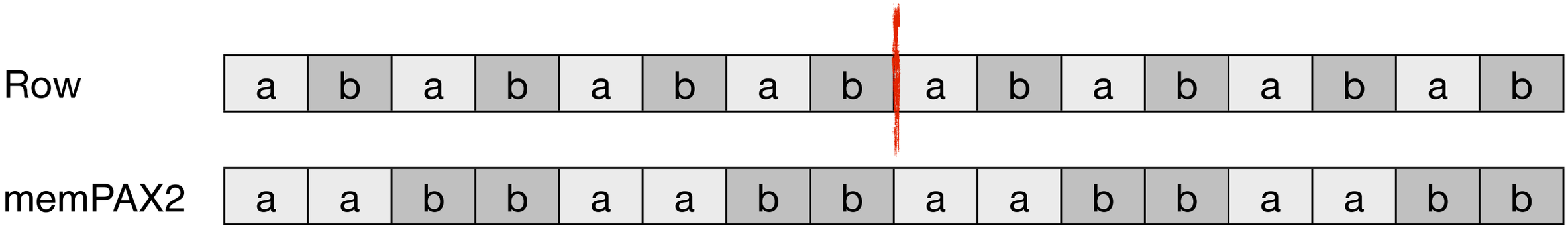


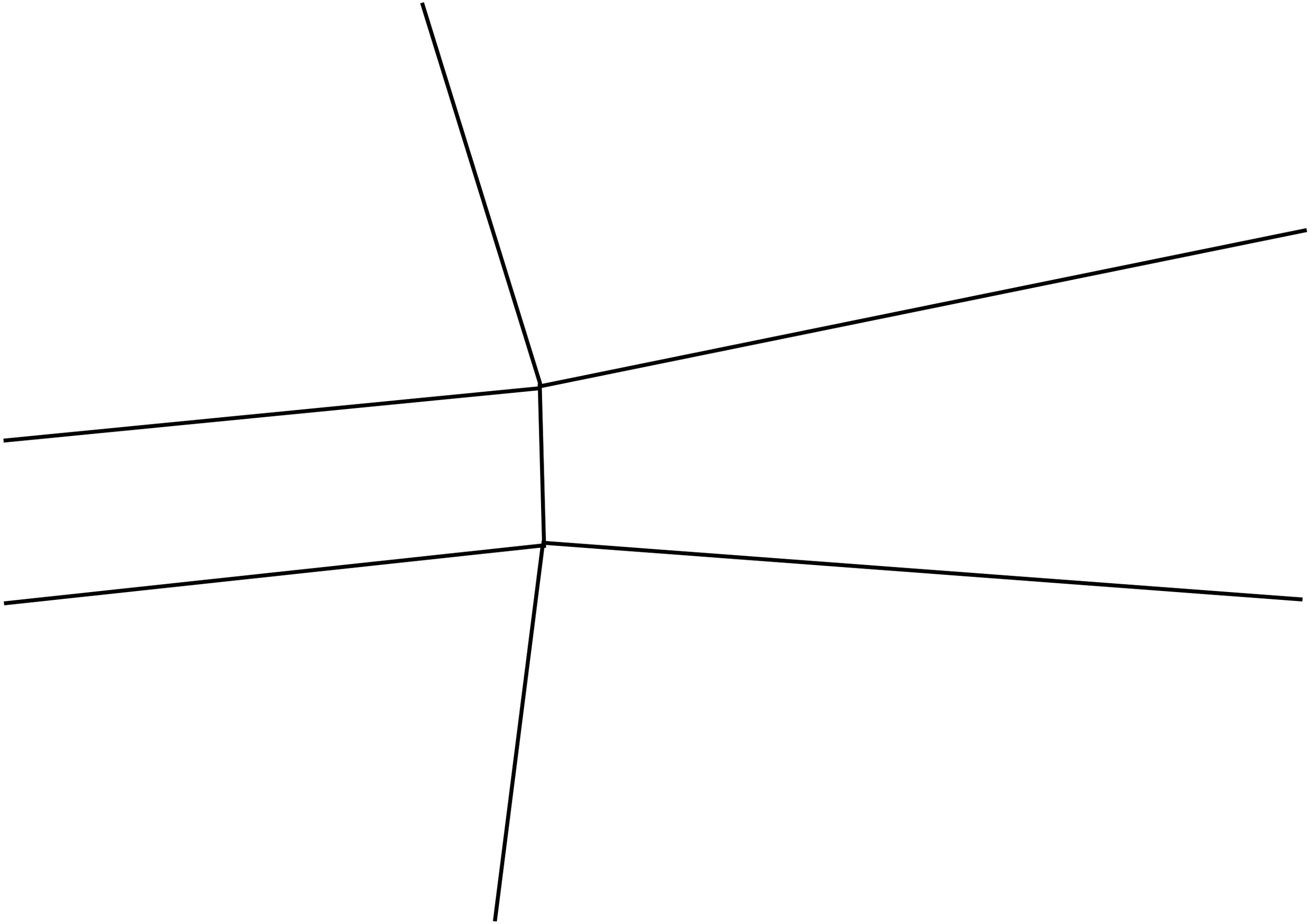

Row
a b a b a b a b a b a b a b a b
memPAX2
a a b b a a b b a a b b a a b b


Column
a a a a a a a a b b b b b b b b

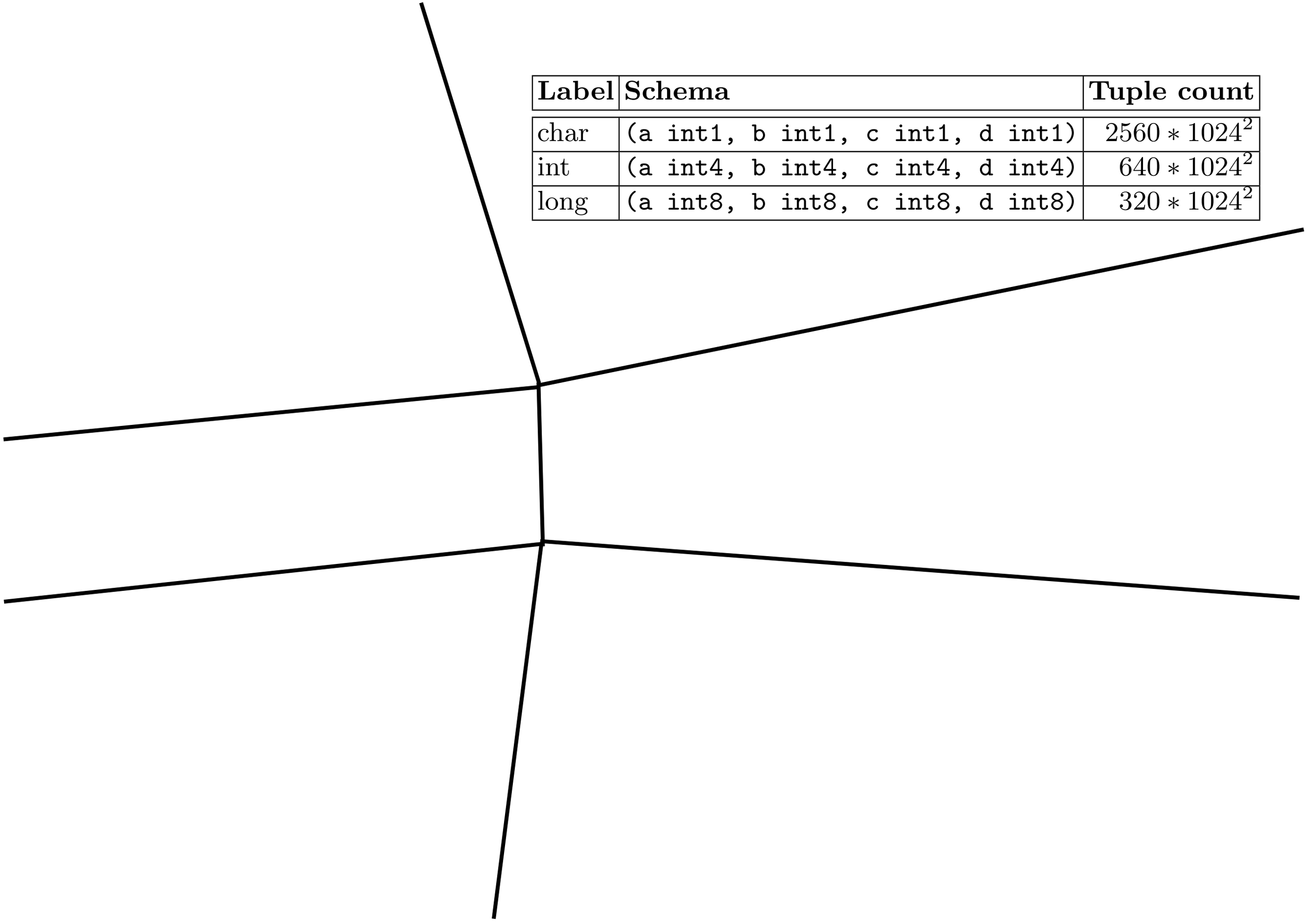

Row
a b a b a b a b a b a b a b a b
memPAX2
a a b b a a b b a a b b a a b b


Column
a a a a a a a a b b b b b b b b


Row
a b a b a b a b a b a b a b a b
memPAX2
a a b b a a b b a a b b a a b b
memPAX4
a a a a b b b b a a a a b b b b
Column
a a a a a a a a b b b b b b b b

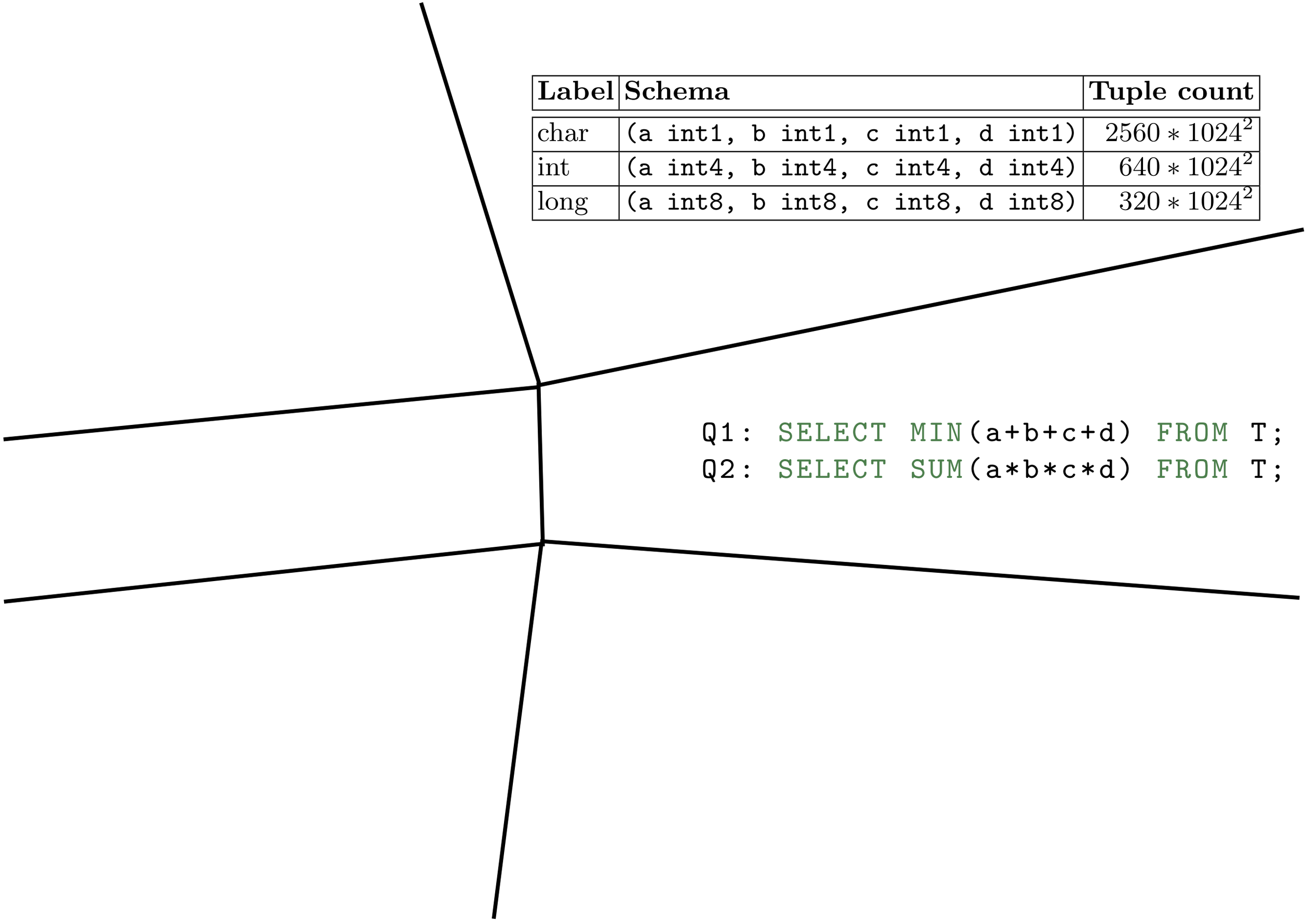


Label	Schema	Tuple count
char	`(a int1, b int1, c int1, d int1)`	$2560 * 1024^2$
int	`(a int4, b int4, c int4, d int4)`	$640 * 1024^2$
long	`(a int8, b int8, c int8, d int8)`	$320 * 1024^2$



Label	Schema	Tuple count
char	`(a int1, b int1, c int1, d int1)`	$2560 * 1024^2$
int	`(a int4, b int4, c int4, d int4)`	$640 * 1024^2$
long	`(a int8, b int8, c int8, d int8)`	$320 * 1024^2$

```
Q1: SELECT MIN(a+b+c+d) FROM T;
Q2: SELECT SUM(a*b*c*d) FROM T;
```



Label	Schema	Tuple count
char	`(a int1, b int1, c int1, d int1)`	$2560 * 1024^2$
int	`(a int4, b int4, c int4, d int4)`	$640 * 1024^2$
long	`(a int8, b int8, c int8, d int8)`	$320 * 1024^2$

```
Q1: SELECT MIN(a+b+c+d) FROM T;
Q2: SELECT SUM(a*b*c*d) FROM T;
```

CPU	Architecture	RAM
Xeon 5150	Core	16 GB DDR2 @ 266 MHz
Xeon X5690	Westmere	192 GB DDR3 @ 1066 MHz
Xeon E5-2407	Sandy Bridge	48 GB DDR3 @ 1333 MHz
Xeon E7-4870 v2	Ivy Bridge	512 GB DDR3 @ 1600 MHz

compiler
gcc
clang
icc

Label	Schema	Tuple count
char	`(a int1, b int1, c int1, d int1)`	$2560 * 1024^2$
int	`(a int4, b int4, c int4, d int4)`	$640 * 1024^2$
long	`(a int8, b int8, c int8, d int8)`	$320 * 1024^2$

```
Q1: SELECT MIN(a+b+c+d) FROM T;
Q2: SELECT SUM(a*b*c*d) FROM T;
```

CPU	Architecture	RAM
Xeon 5150	Core	16 GB DDR2 @ 266 MHz
Xeon X5690	Westmere	192 GB DDR3 @ 1066 MHz
Xeon E5-2407	Sandy Bridge	48 GB DDR3 @ 1333 MHz
Xeon E7-4870 v2	Ivy Bridge	512 GB DDR3 @ 1600 MHz

compiler
gcc
clang
icc

Label	Schema	Tuple count
char	(a int1, b int1, c int1, d int1)	$2560 * 1024^2$
int	(a int4, b int4, c int4, d int4)	$640 * 1024^2$
long	(a int8, b int8, c int8, d int8)	$320 * 1024^2$

```
Q1: SELECT MIN(a+b+c+d) FROM T;
Q2: SELECT SUM(a*b*c*d) FROM T;
```

-O1, -O2, and -O3.

CPU	Architecture	RAM
Xeon 5150	Core	16 GB DDR2 @ 266 MHz
Xeon X5690	Westmere	192 GB DDR3 @ 1066 MHz
Xeon E5-2407	Sandy Bridge	48 GB DDR3 @ 1333 MHz
Xeon E7-4870 v2	Ivy Bridge	512 GB DDR3 @ 1600 MHz

compiler
gcc
clang
icc

Label	Schema	Tuple count
char	(a int1, b int1, c int1, d int1)	$2560 * 1024^2$
int	(a int4, b int4, c int4, d int4)	$640 * 1024^2$
long	(a int8, b int8, c int8, d int8)	$320 * 1024^2$

```
Q1: SELECT MIN(a+b+c+d) FROM T;
Q2: SELECT SUM(a*b*c*d) FROM T;
```

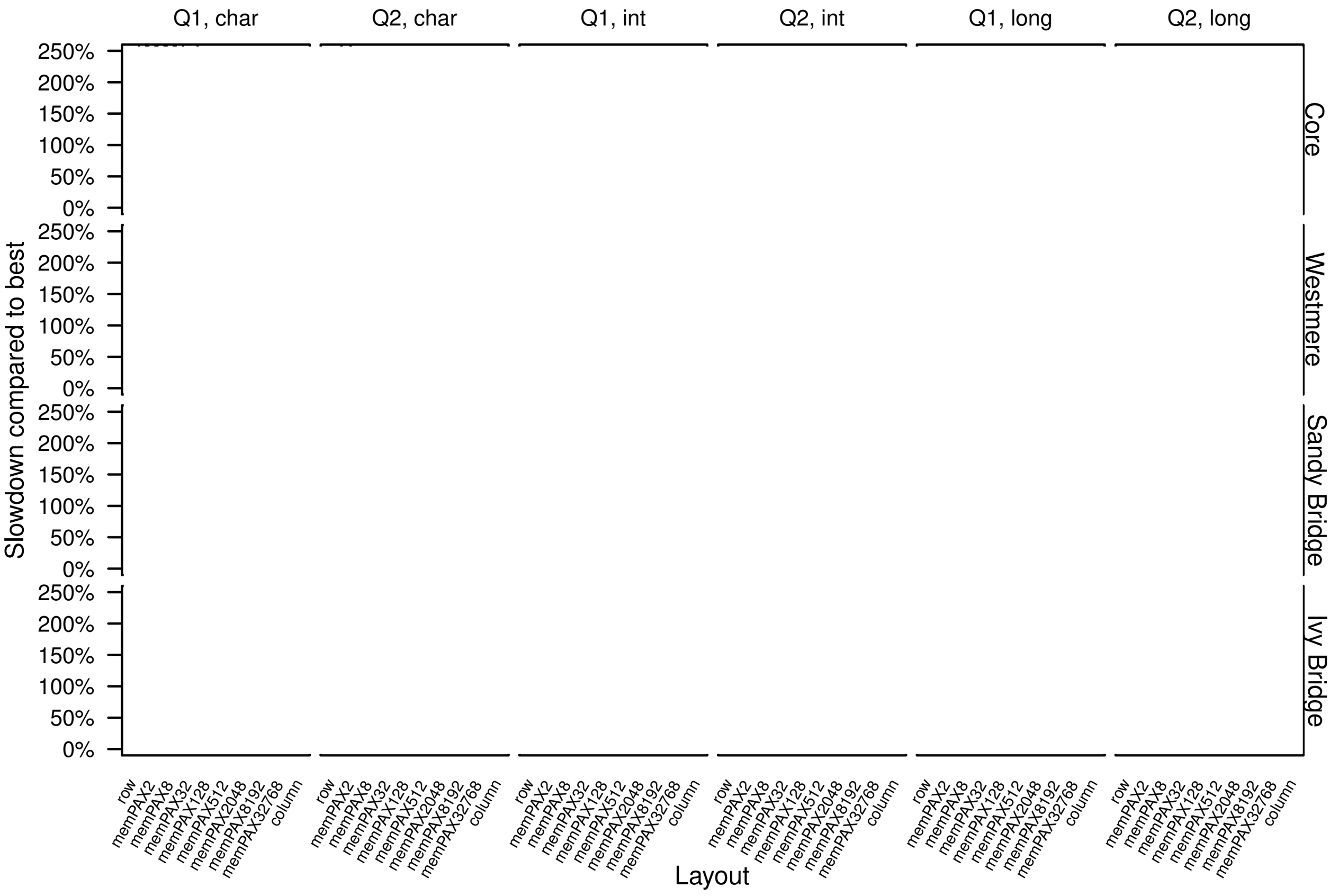
-O1, -O2, and -O3.

compile-time
vs.
runtime layouts

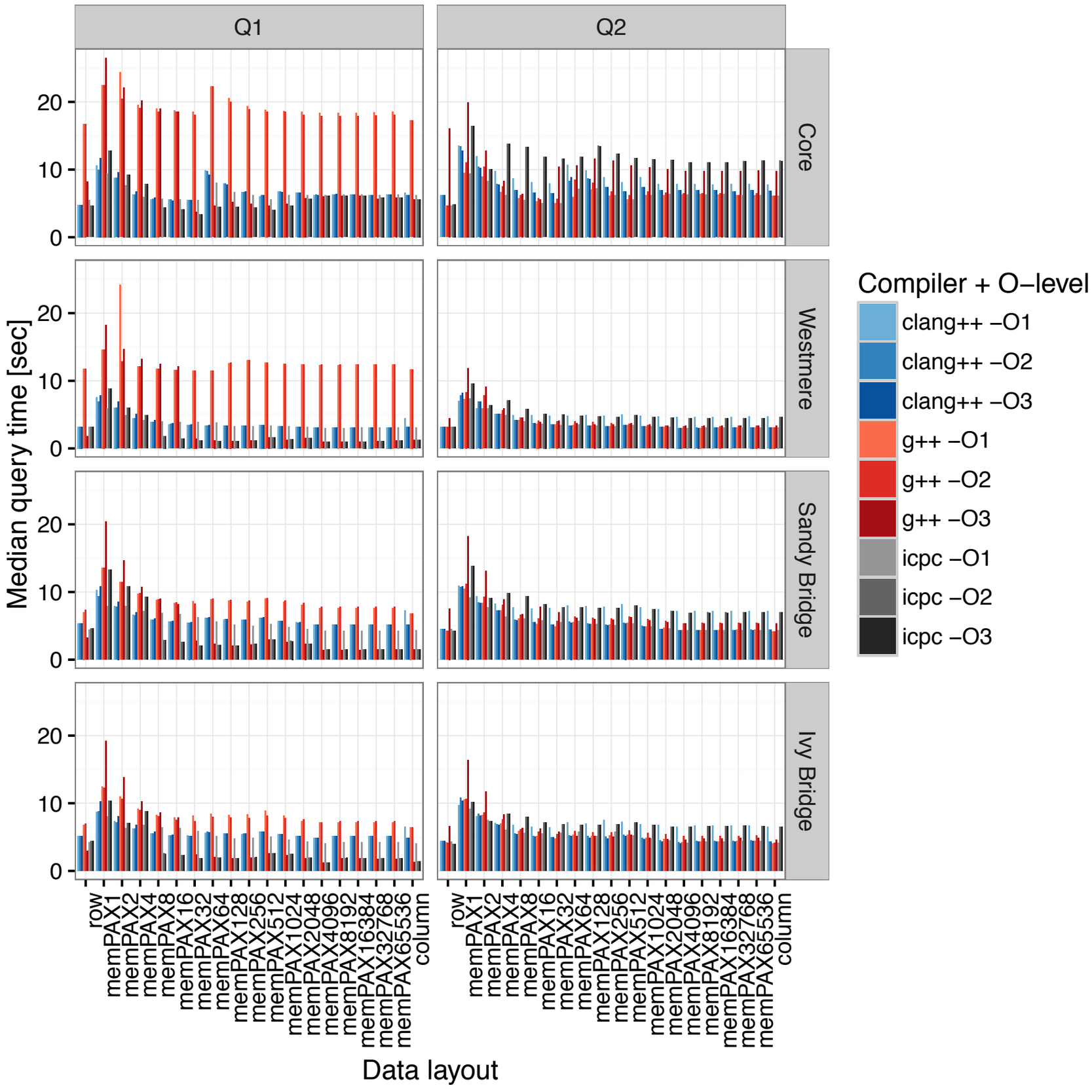
CPU	Architecture	RAM
Xeon 5150	Core	16 GB DDR2 @ 266 MHz
Xeon X5690	Westmere	192 GB DDR3 @ 1066 MHz
Xeon E5-2407	Sandy Bridge	48 GB DDR3 @ 1333 MHz
Xeon E7-4870 v2	Ivy Bridge	512 GB DDR3 @ 1600 MHz

Experimental setup

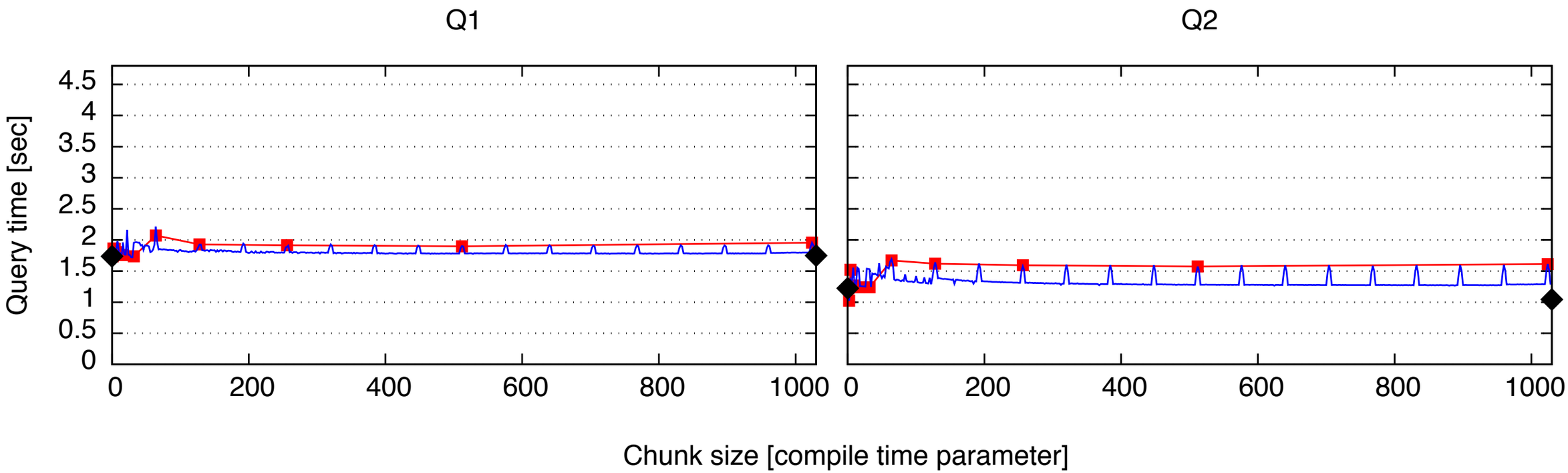
- memory resident tables (shm)
- hand-coded queries implemented in C++
- single-threaded
- process pinned to core
- 3 warm-up runs, 5 warm runs


Q1, char
Q2, char
Q1, int
Q2, int
Q1, long
Q2, long
Core
Westmere
Sandy Bridge
Ivy Bridge
Slowdown compared to best
250%
200%
150%
100%
50%
0%
row
memPAX2
memPAX8
memPAX32
memPAX128
memPAX512
memPAX2048
memPAX8192
memPAX32768
column
Layout

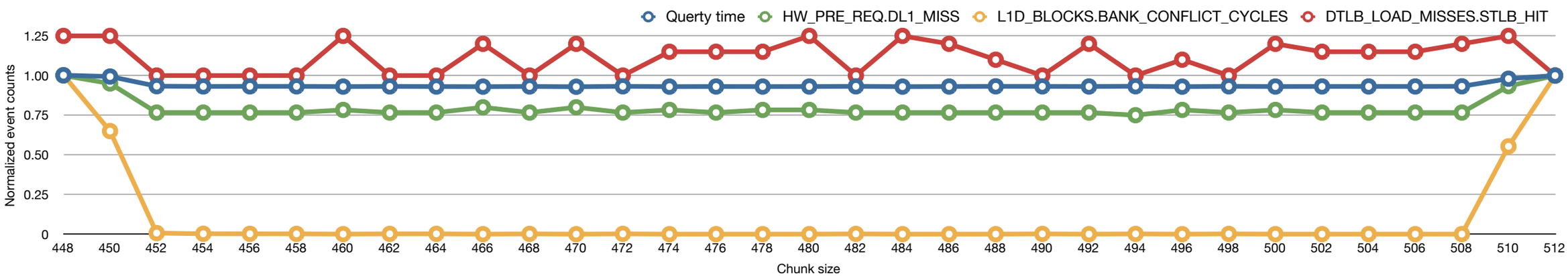
Q1, char
Q2, char
Q1, int
Q2, int
Q1, long
Q2, long
Core
Westmere
Sandy Bridge
Ivy Bridge
Slowdown compared to best
250%
200%
150%
100%
50%
0%
row
memPAX2
memPAX8
memPAX32
memPAX128
memPAX512
memPAX2048
memPAX8192
memPAX32768
column
Layout



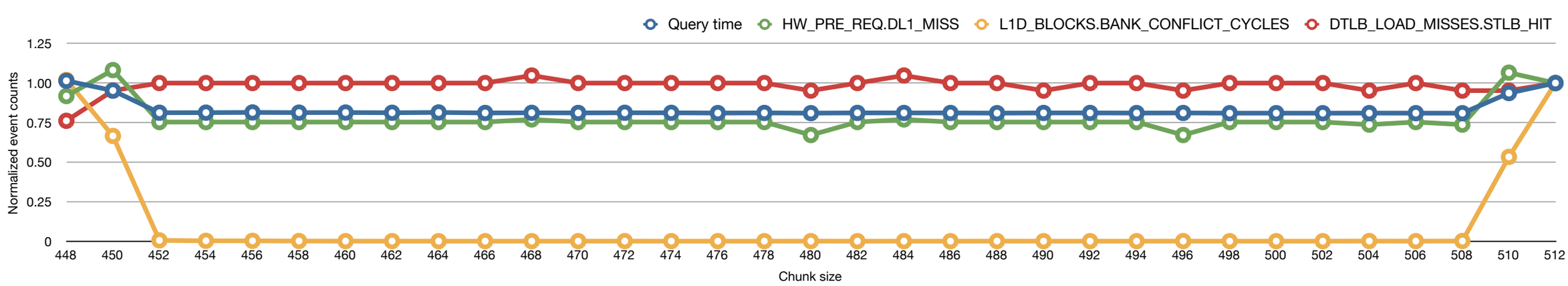
. Query times on the `char` schema across compiler setups and runtime-layouts in our micro benchmarks.



Query times of Q1 and Q2 executed on Sandy Bridge on `char` fields, compiled with `g++ -O2`, and the chunk size being provided at compile time.

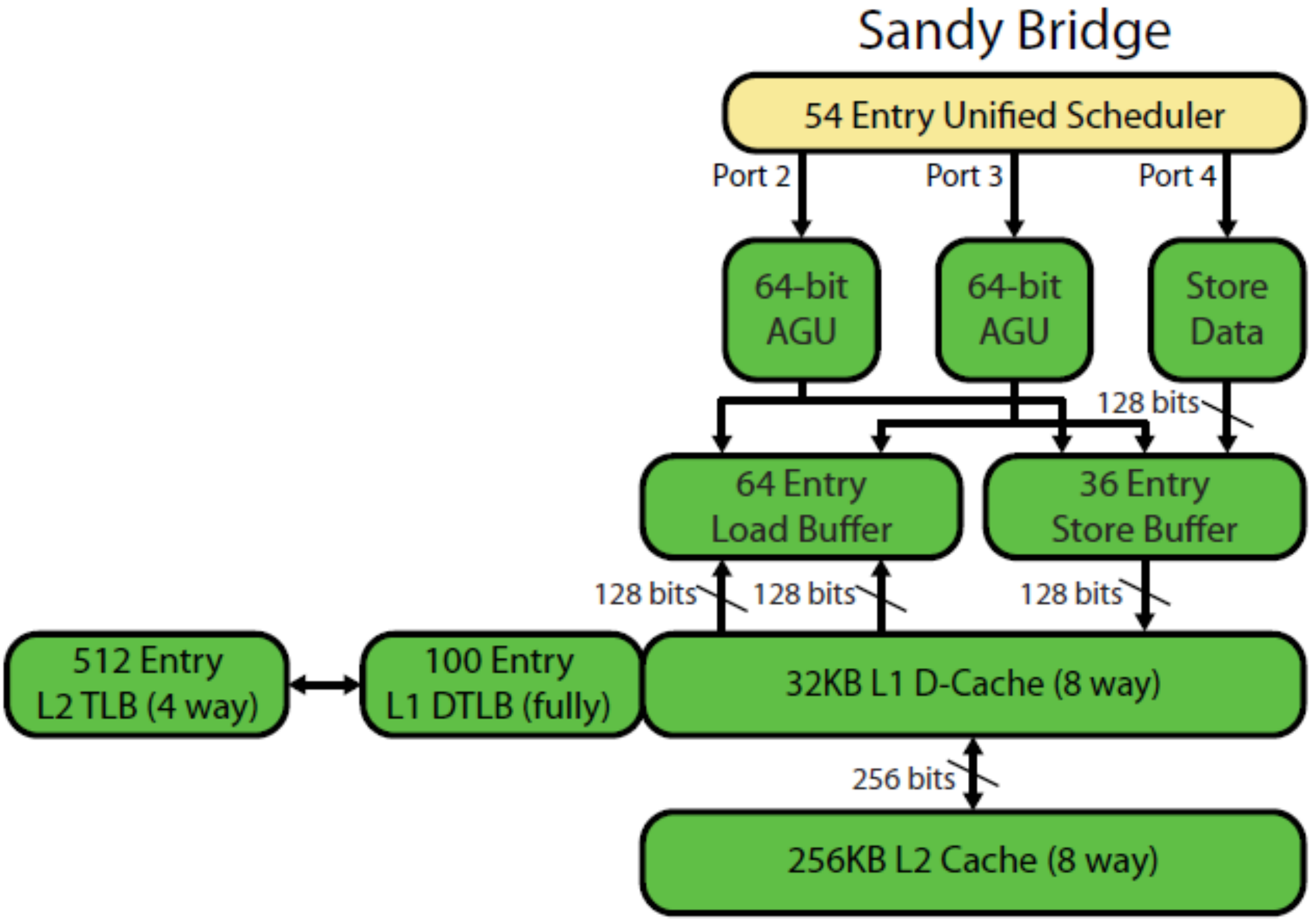


(a) Q1

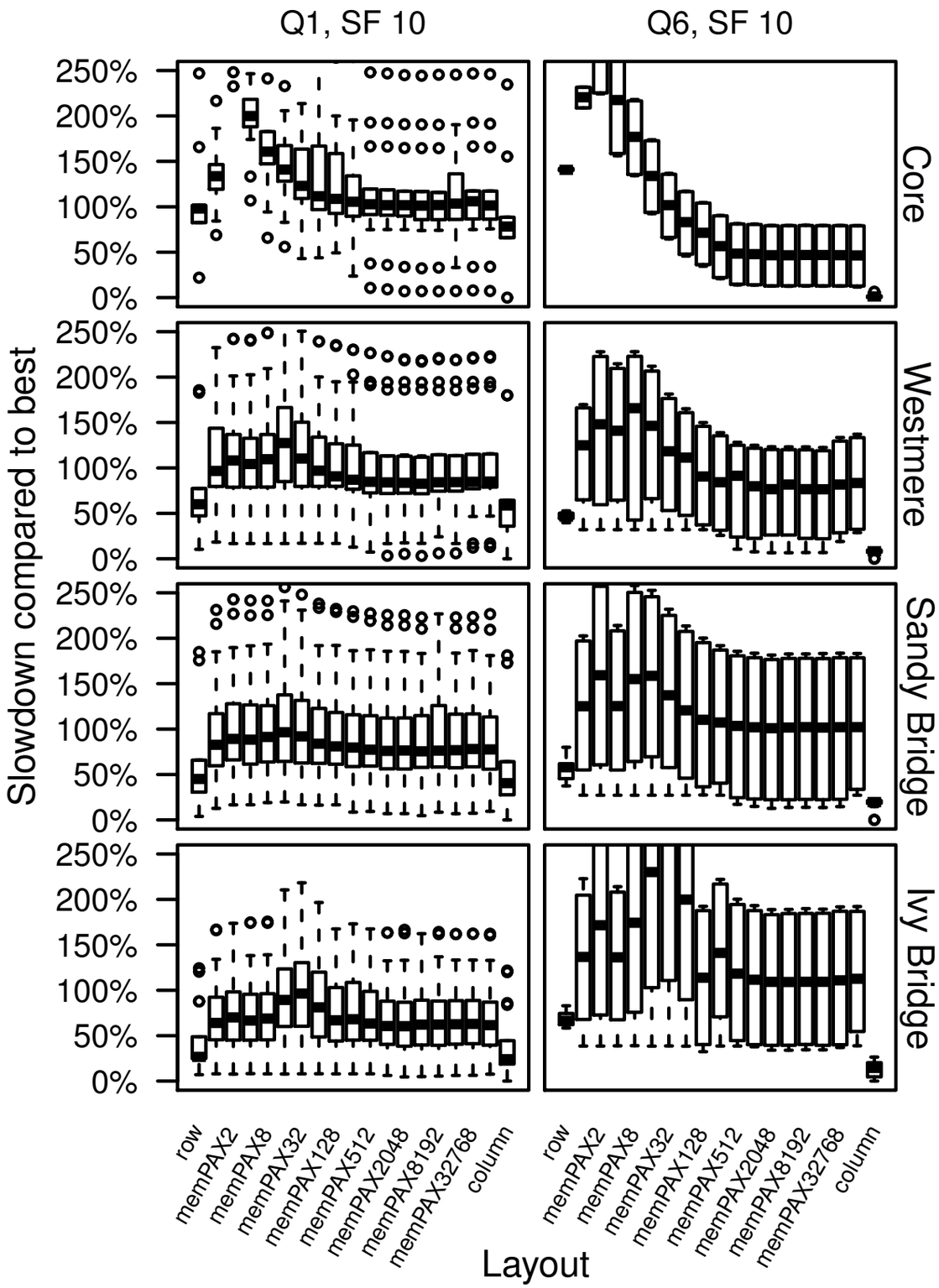


(b) Q2

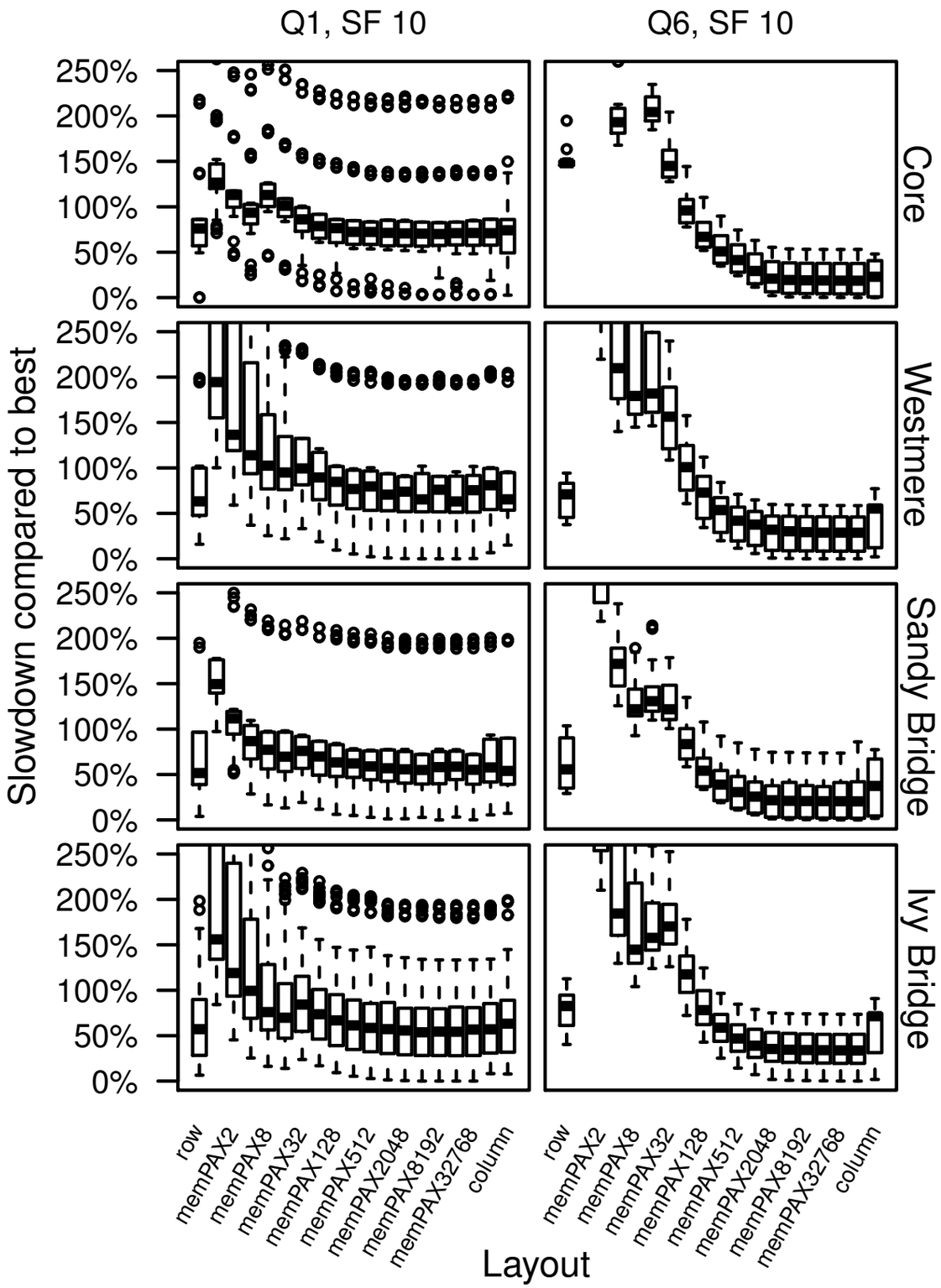
PMU events of Q1 and Q2 executed on Sandy Bridge on `char` fields, compiled with `g++ -O2`, for chunk sizes in $\{448, 450, \ldots, 510, 512\}$.


Sandy Bridge
54 Entry Unified Scheduler
Port 2
Port 3
Port 4
64-bit AGU
64-bit AGU
Store Data
128 bits
64 Entry Load Buffer
36 Entry Store Buffer
128 bits
128 bits
128 bits
512 Entry L2 TLB (4 way)
100 Entry L1 DTLB (fully)
32KB L1 D-Cache (8 way)
256 bits
256KB L2 Cache (8 way)

8 banks



(a) Over uncompressed tables



(b) Over compressed tables

Runtime fragility of the various data layouts for TPC-H queries

Conclusion

- Use confidence intervals or the median
- The compiler is *not* yet another abstraction layer
- Beware of low-level CPU bus/port/etc. contention in new hardware