

# Comparison of job resource consumption at different cache states

The following is a test that shows the effects of the Linux filesystem cache being „hot“ or „cold“ (filled with data needed for the job) and the PostgreSQL shared\_buffers caches being „hot“ or „cold“ on job run-time and resource usage profiles.

In particular the CPU load, the ratio between system and user CPU and the storage IO load consumption profiles change when caches are cold versus hot.

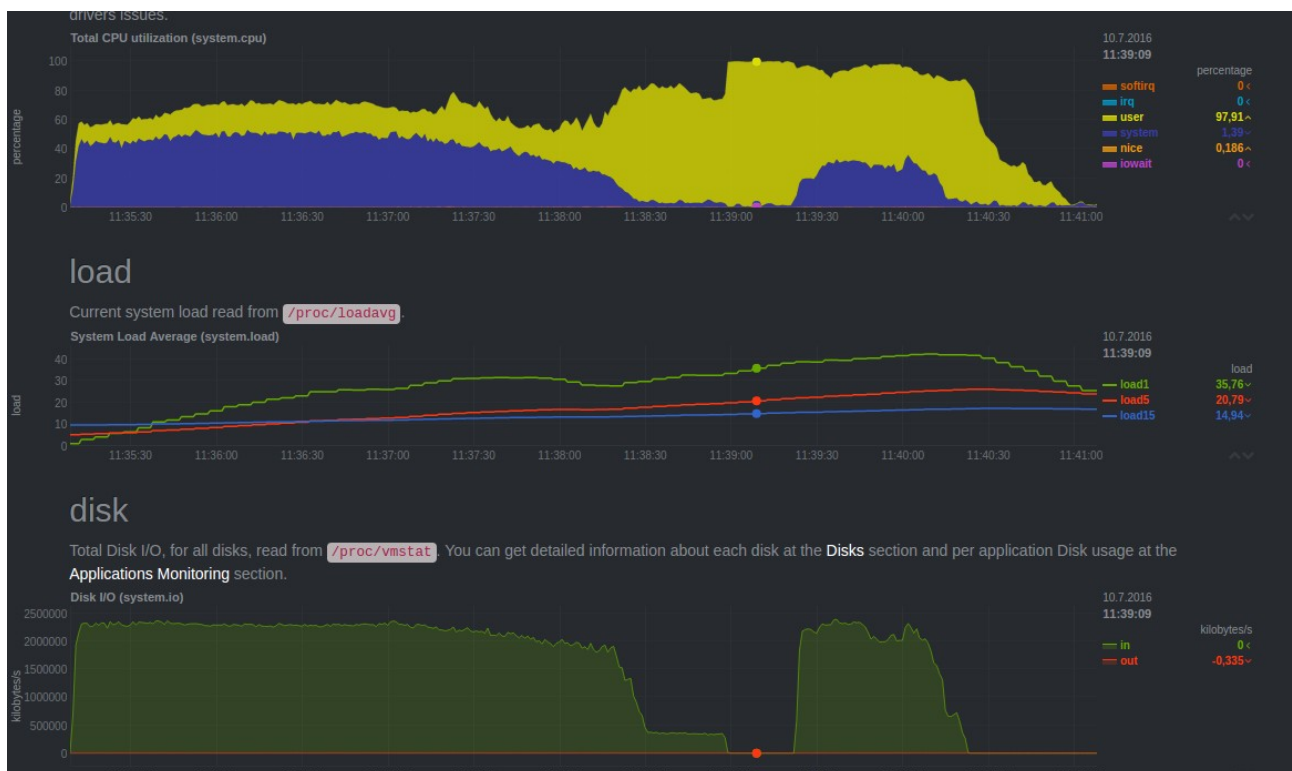
## Test Job

The test job used can be found in Git here:

[<GIT-RA>/adr/services/svc\\_sqlbalancer/examples/rating\\_ereignis\\_summary.sql](https://git-ra.adr.services/svc_sqlbalancer/examples/rating_ereignis_summary.sql)

## PG-Cold / FS-Cold

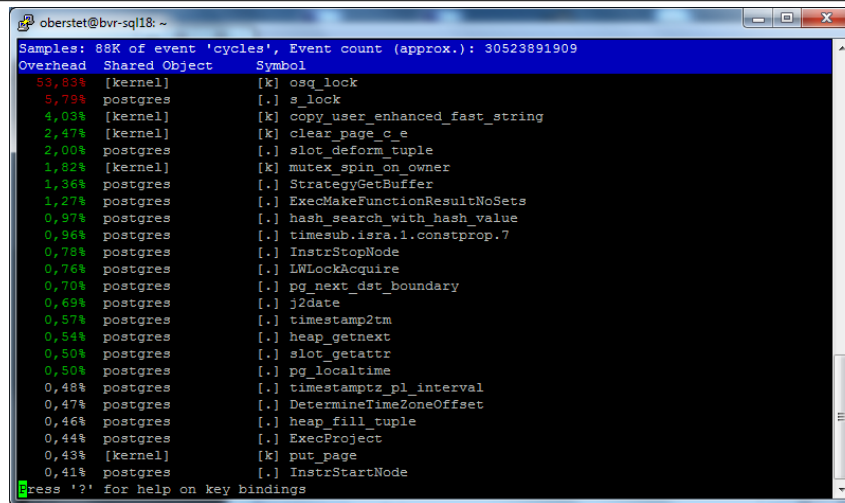
Here is the CPU/disk load during a complete run of the test job with PostgreSQL cache cold and Filesystem cache cold:



With both the FS and PG caches cold, there are phases with massive storage read IO (> 2.3GB/s). Here, data is read from the NVMe's and brought into the Linux FS and further mapped into the PG shared buffers. During these phases, the system CPU load is also significant. This is expected, as storage IO will run kernel (FS and driver) code.

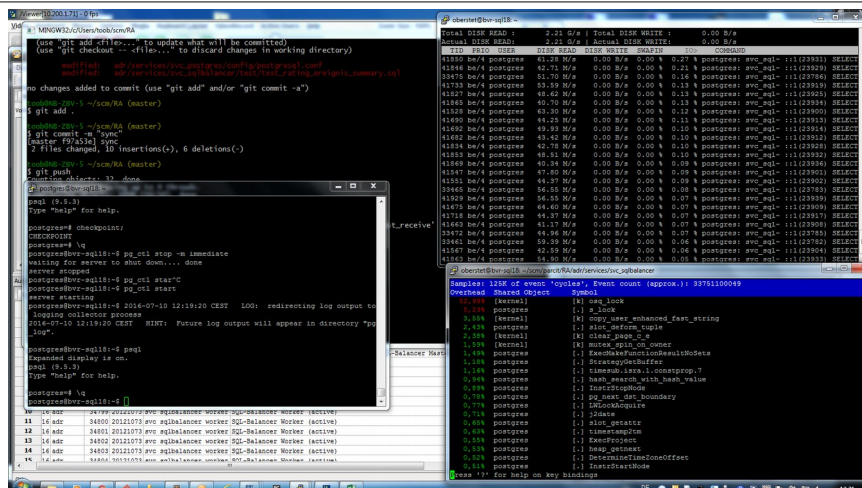
Here is a snapshot of Linux perf top monitoring a PG background process of a SQL balancer worker running a work unit of the test job:

High system load:



```
oberstet@bvr-sql18: ~$ perf top
Samples: 88K of event 'cycles', Event count (approx.): 30523891909
Overhead Shared Object Symbol
53.83% [kernel] [k] osq_lock
5.79% postgres [.] s_lock
4.03% [kernel] [k] copy_user_enhanced_fast_string
2.47% [kernel] [k] clear_page_c_e
2.00% postgres [.] slot_deform_tuple
1.82% [kernel] [k] mutex_spin_on_owner
1.36% postgres [.] StrategyGetBuffer
1.27% postgres [.] ExecMakeFunctionResultNoSets
0.97% postgres [.] hash_search_with_hash_value
0.96% postgres [.] timesub.isra.1.constprop.7
0.78% postgres [.] InstrStopNode
0.76% postgres [.] LWLockAcquire
0.70% postgres [.] pg_next_dst_boundary
0.69% postgres [.] j2date
0.57% postgres [.] timestamp2tm
0.54% postgres [.] heap_getnext
0.50% postgres [.] slot_getattr
0.50% postgres [.] pg_localtime
0.48% postgres [.] timestamptz_pl_interval
0.47% postgres [.] DetermineTimeZoneOffset
0.46% postgres [.] heap_fill_tuple
0.44% postgres [.] ExecProject
0.43% [kernel] [k] put_page
0.41% postgres [.] InstrStartNode
Press '?' for help on key bindings
```

High storage IO:



```
oberstet@bvr-sql18: ~$ perf top
Samples: 128K of event 'cycles', Event count (approx.): 5375100049
Overhead Shared Object Symbol
3.89% [kernel] [k] s_lock
3.89% [kernel] [k] copy_user_enhanced_fast_string
3.89% postgres [.] slot_deform_tuple
3.89% [kernel] [k] clear_page_c_e
3.89% [kernel] [k] mutex_spin_on_owner
3.89% postgres [.] ExecMakeFunctionResultNoSets
3.89% postgres [.] timesub.isra.1.constprop.7
3.89% postgres [.] hash_search_with_hash_value
3.89% postgres [.] InstrStopNode
3.89% postgres [.] pg_next_dst_boundary
3.89% postgres [.] j2date
3.89% postgres [.] timestamp2tm
3.89% postgres [.] heap_getnext
3.89% postgres [.] slot_getattr
3.89% postgres [.] pg_localtime
3.89% postgres [.] timestamptz_pl_interval
3.89% postgres [.] DetermineTimeZoneOffset
3.89% postgres [.] heap_fill_tuple
3.89% postgres [.] ExecProject
3.89% [kernel] [k] put_page
3.89% postgres [.] InstrStartNode
Press '?' for help on key bindings
```

## PG-Cold / FS-Hot

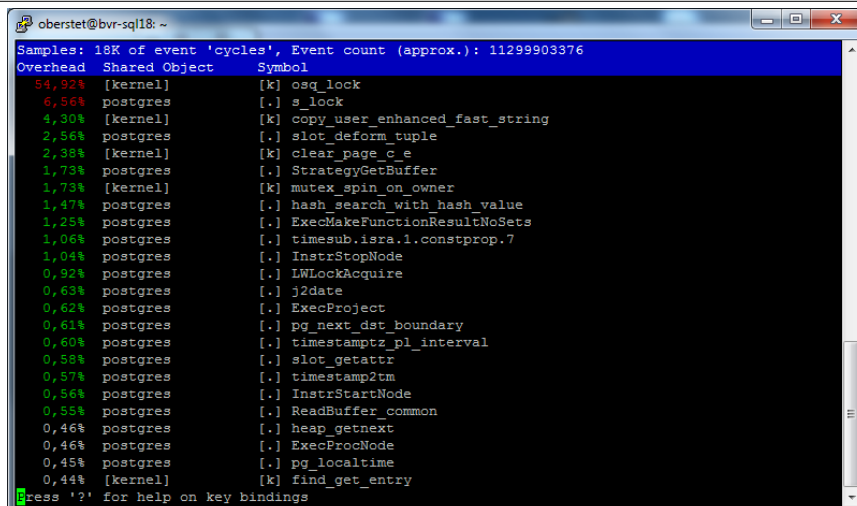
Here is the CPU/disk load during a complete run of the test job with PostgreSQL cache cold and Filesystem cache hot:



The CPU load total and breakdown between user/system CPU look nearly the same as with cold FS caches. However, there is no IO at all. This makes sense as data is fully cached in FS caches already. And it demonstrates that PG will first need to bring data into PG caches for processing.

Here is a snapshot of Linux perf top monitoring a PG background process of a SQL balancer worker running a work unit of the test job:

#### Phase 1: bringing data in PG caches



#### Phase 2: crunching data

oberstet@bvr-sql18: ~

Samples: 431K of event 'cycles', Event count (approx.): 25678782961

Overhead	Shared Object	Symbol
29,26%	postgres	[.] _bt_checkkeys
25,08%	[kernel]	[k] osq_lock
5,16%	postgres	[.] s_lock
4,94%	postgres	[.] hash_search_with_hash_value
4,32%	[kernel]	[k] copy_user_enhanced_fast_string
4,23%	postgres	[.] FunctionCall2Coll
3,76%	postgres	[.] LWLockAcquire
1,89%	[kernel]	[k] mutex_spin_on_owner
1,84%	postgres	[.] StrategyGetBuffer
1,81%	[kernel]	[k] clear_page_c_e
1,39%	postgres	[.] PinBuffer
1,34%	postgres	[.] _bt_readpage
1,16%	postgres	[.] UnlockReleaseBuffer
1,14%	postgres	[.] LWLockRelease
0,95%	postgres	[.] timestamp_lt
0,92%	postgres	[.] timestamp_ge
0,74%	postgres	[.] ReadBuffer_common
0,58%	postgres	[.] _bt_steppage
0,43%	[kernel]	[k] put_page
0,37%	postgres	[.] _bt_checkpoint
0,36%	[kernel]	[k] update_ofs_shares
0,35%	[kernel]	[k] effective_load.isra.40
0,30%	[kernel]	[k] _raw_spin_lock
0,27%	postgres	[.] slot_deform_tuple

Press '?' for help on key bindings

oberstet@bvr-sql18: ~

Samples: 889K of event 'cycles', Event count (approx.): 52628687571

Overhead	Shared Object	Symbol
27,34%	postgres	[.] pg_detoast_datum_packed
19,32%	postgres	[.] _bt_checkkeys
12,12%	libc-2.21.so	[.] __strcoll_l
6,02%	postgres	[.] ExecStoreMinimalTuple
5,42%	postgres	[.] qsort_ssup
3,28%	postgres	[.] FunctionCall2Coll
3,09%	postgres	[.] varstr_cmp
2,86%	postgres	[.] bpcharcmp
1,92%	postgres	[.] slot_deform_tuple
1,62%	postgres	[.] hash_search_with_hash_value
1,40%	postgres	[.] comparison_shim
1,28%	postgres	[.] LWLockAcquire
1,04%	postgres	[.] btint8fastcmp
0,80%	postgres	[.] _bt_readpage
0,75%	libc-2.21.so	[.] __strcoll_l
0,62%	postgres	[.] tuplesort_gettupleslot
0,59%	postgres	[.] lc_collate_is_c
0,58%	postgres	[.] ExecProject
0,57%	postgres	[.] advance_aggregates
0,56%	postgres	[.] _bt_steppage
0,54%	postgres	[.] timestamp_ge
0,53%	postgres	[.] timestamp_lt
0,53%	postgres	[.] slot_getattr
0,43%	postgres	[.] bpchareq

Press '?' for help on key bindings

### Phase 3: storing results

oberstet@bvr-sql18: ~

Samples: 953K of event 'cycles', Event count (approx.): 53442218624

Overhead	Shared Object	Symbol
22,05%	postgres	[.] ExecStoreMinimalTuple
15,44%	postgres	[.] qsort_ssup
9,40%	postgres	[.] pg_detoast_datum_packed
6,83%	postgres	[.] slot_deform_tuple
6,54%	postgres	[.] _bt_checkkeys
6,27%	postgres	[.] btint8fastcmp
4,10%	libc-2.21.so	[.] __strcoll_l
2,90%	postgres	[.] FunctionCall2Coll
1,97%	postgres	[.] tuplesort_gettupleslot
1,67%	postgres	[.] ExecProject
1,58%	postgres	[.] advance_aggregates
1,46%	postgres	[.] tuplesort_getdatum
1,38%	postgres	[.] slot_getattr
1,11%	postgres	[.] bpchareq
1,05%	postgres	[.] varstr_cmp
1,00%	postgres	[.] execTuplesMatch
0,97%	postgres	[.] bpcharcmp
0,96%	postgres	[.] puttuple_common
0,90%	postgres	[.] advance_transition_function
0,90%	postgres	[.] InstrStopNode
0,65%	postgres	[.] InstrStartNode
0,62%	postgres	[.] slot_getsomeattrs
0,58%	[kernel]	[k] clear_page_c_e
0,57%	postgres	[.] ExecProcNode

Press '?' for help on key bindings

## PG-Hot / FS-Hot

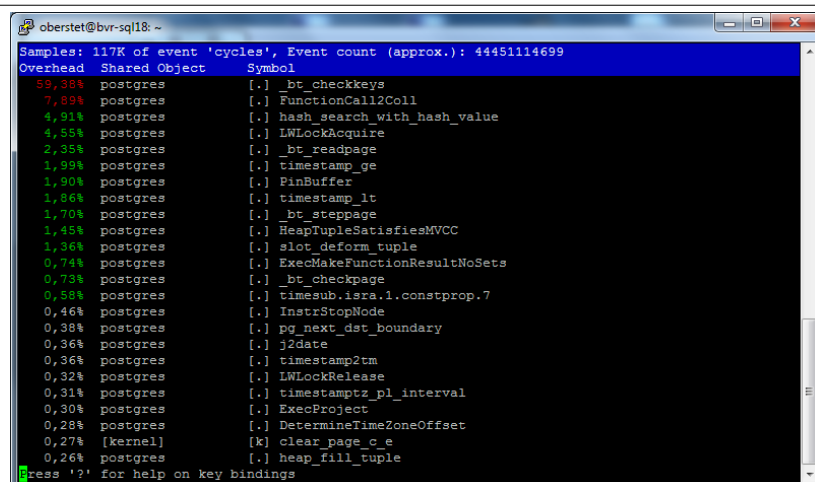
Here is the CPU/disk load during a complete run of the test job with PostgreSQL cache hot and Filesystem cache hot:



With both the FS and PG caches hot, time is spent almost exclusively in PG code. The specific functions within PG consuming the most time is changing of the run-time of the test job and depend on the phase in which the job.

Here is a snapshot of Linux `perf top` monitoring a PG background process of a SQL balancer worker running a work unit of the test job:

### Phase 1: accessing source data



### Phase 2: computing result set

```
oberstet@bvr-sql18: ~  
Samples: 458K of event 'cycles', Event count (approx.): 52232857789  
Overhead Shared Object Symbol  
14,98% postgres [.] pg_detoast_datum_packed  
14,50% postgres [.] qsort_ssup  
13,39% postgres [.] ExecStoreMinimalTuple  
8,66% postgres [.] _bt_checkkeys  
5,52% libc-2.21.so [.] __strcoll_l  
5,40% postgres [.] bint8fastcmp  
4,66% postgres [.] slot_deform_tuple  
3,04% postgres [.] FunctionCall2Coll  
1,50% postgres [.] ExecProject  
1,44% postgres [.] varstr_cmp  
1,36% postgres [.] bpcharcmp  
1,35% postgres [.] slot_getattr  
1,28% postgres [.] advance_aggregates  
1,28% postgres [.] tuplesort_gettupleslot  
1,06% postgres [.] execTuplesMatch  
1,05% postgres [.] advance_transition_function  
1,00% postgres [.] bpchareq  
0,97% postgres [.] tuplesort_getdatum  
0,84% postgres [.] InstrStopNode  
0,82% [kernel] [k] copy_page_rep  
0,71% postgres [.] puttuple_common  
0,68% postgres [.] comparison_shim  
0,67% postgres [.] InstrStartNode  
0,58% postgres [.] ExecClearTuple  
Press '?' for help on key bindings
```

### Phase 3: storing results

```
oberstet@bvr-sql18: ~  
Samples: 496K of event 'cycles', Event count (approx.): 39208347296  
Overhead Shared Object Symbol  
19,34% postgres [.] ExecStoreMinimalTuple  
17,11% postgres [.] qsort_ssup  
7,93% postgres [.] pg_detoast_datum_packed  
6,93% postgres [.] bint8fastcmp  
6,59% postgres [.] slot_deform_tuple  
4,53% postgres [.] _bt_checkkeys  
2,91% postgres [.] FunctionCall2Coll  
2,89% libc-2.21.so [.] __strcoll_l  
2,31% postgres [.] advance_transition_function  
1,80% postgres [.] ExecProject  
1,58% postgres [.] slot_getattr  
1,48% postgres [.] tuplesort_gettupleslot  
1,47% postgres [.] advance_aggregates  
1,27% postgres [.] tuplesort_getdatum  
1,22% postgres [.] execTuplesMatch  
1,20% postgres [.] bpchareq  
0,99% postgres [.] InstrStopNode  
0,87% postgres [.] puttuple_common  
0,80% [kernel] [k] copy_page_rep  
0,75% postgres [.] InstrStartNode  
0,75% postgres [.] varstr_cmp  
0,74% postgres [.] finalize_aggregates  
0,71% postgres [.] bpcharcmp  
0,67% postgres [.] ExecClearTuple  
Press '?' for help on key bindings
```

**Conclusion:** With data fully cached in PG caches, the whole CPU available on the machine being spent almost exclusively within PG code, not kernel (or other) code means, that any tuning effort can focus tuning at the SQL level, rather than system (Linux) or storage (FS, hardware).