EDB Postgres Advanced Server on IBM zSystems

Performance Tuning Guide

Marc Beyerle (marc.beyerle@de.ibm.com)
Senior Java Performance Engineer, IBM Mainframe Specialist

Document version: 1.0
Document date: 2023-04-17

Notices and disclaimers

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers, *continued*

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and FICON, IBM FlashSystem, IBM Z, IBM z16, IBM zSystems, POWER, and z16 are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names and logos might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml

Agenda

- Introduction to EDB Postgres Advanced Server
- Performance measurement and tuning approach
- Observations and recommendations
- Summary

About EDB Postgres Advanced Server

- Relational database management system based on Open Source PostgreSQL®, which in turn claims to be *"The World's Most Advanced Open Source Relational Database"* according to their website

- PostgreSQL is #4 on https://db-engines.com/en/ranking

- *EDB® Postgres® Advanced Server* (EPAS) adds ***extended functionality*** to Open Source PostgreSQL
  - Enhanced ***Oracle® compatibility*** features
  - Database administration features and tools
  - EDB Resource Manager

- Important for enterprise-level customers: professional ***service & support***

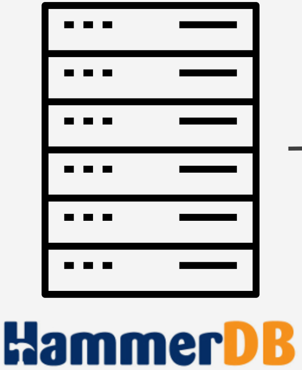
About EDB Postgres Advanced Server, *continued*

- *EnterpriseDB*® (EDB) is a strategic global Open Source ***database partner*** for IBM®
  - EDB is the largest dedicated PostgreSQL company
  - Working with EDB since 2008 on EDB Postgres Plus® Advanced Server for Linux® on IBM POWER®
- EPAS is available for Linux on IBM zSystems® and IBM® LinuxONE
  - IBM United States Software Announcement 222-173: see this link
- On top of this tuning guide, we also performed a ***competitive comparison*** against Intel® x86
  - For details, see the official IBM z16™ Proof Points slide deck

Agenda

- Introduction to EDB Postgres Advanced Server
- Performance measurement and tuning approach
- Observations and recommendations
- Summary

High-level environment setup



Switch

FCP

- IBM FlashSystem® 9200
- 8 x 256 GiB SCSI LUNs for EDB data
- 8 x 64 GiB SCSI LUNs for EDB logs
- 1 x 512 GiB SCSI LUN for EDB backup

- Intel x86, Cascade Lake generation (Intel Xeon® Gold 5218 CPU @ 2.30GHz)
- 32 CPUs, Hyper-Threading
- 256 GiB mem
- 1 x Intel Ethernet Connection X722 for 10GbE SFP+
- 1 x Mellanox® Technologies MT27700 Family [ConnectX®-4]

- z16™ @ 5.2 GHz
- 24 IFLs, SMT-2
- 128 GiB mem
- 1 x EAV DASD for root filesystem
- 1 x OSA-Express7S 10 GbE LR
- 1 x RoCE Express3 25 GbE SR
- 8 x FICON® Express 32SA SX

Approach

- Typically, I start with a small ***sandbox environment***, in order to get familiar with the individual software components (OS, database management system, load driver) and experiment with parameters and settings
- Based on my ***previous experience*** with Open Source PostgreSQL – see this link on IBM Developer – I set up a ***large test environment*** right from the beginning
- Applied all of the tuning recommendations that I discovered back in 2017 (see above IBM Developer link) one at a time and they all turned out to be ***valid still*** (some needed adjustment)
- Nevertheless, found ***additional*** tuning possibilities by research and run-time analysis of EPAS
- Additional difference: started with quite a ***decent baseline configuration*** right from the beginning
  - RoCE Express adapters for the network, IBM FS9200 storage server with NVMe®-based disks, etc.

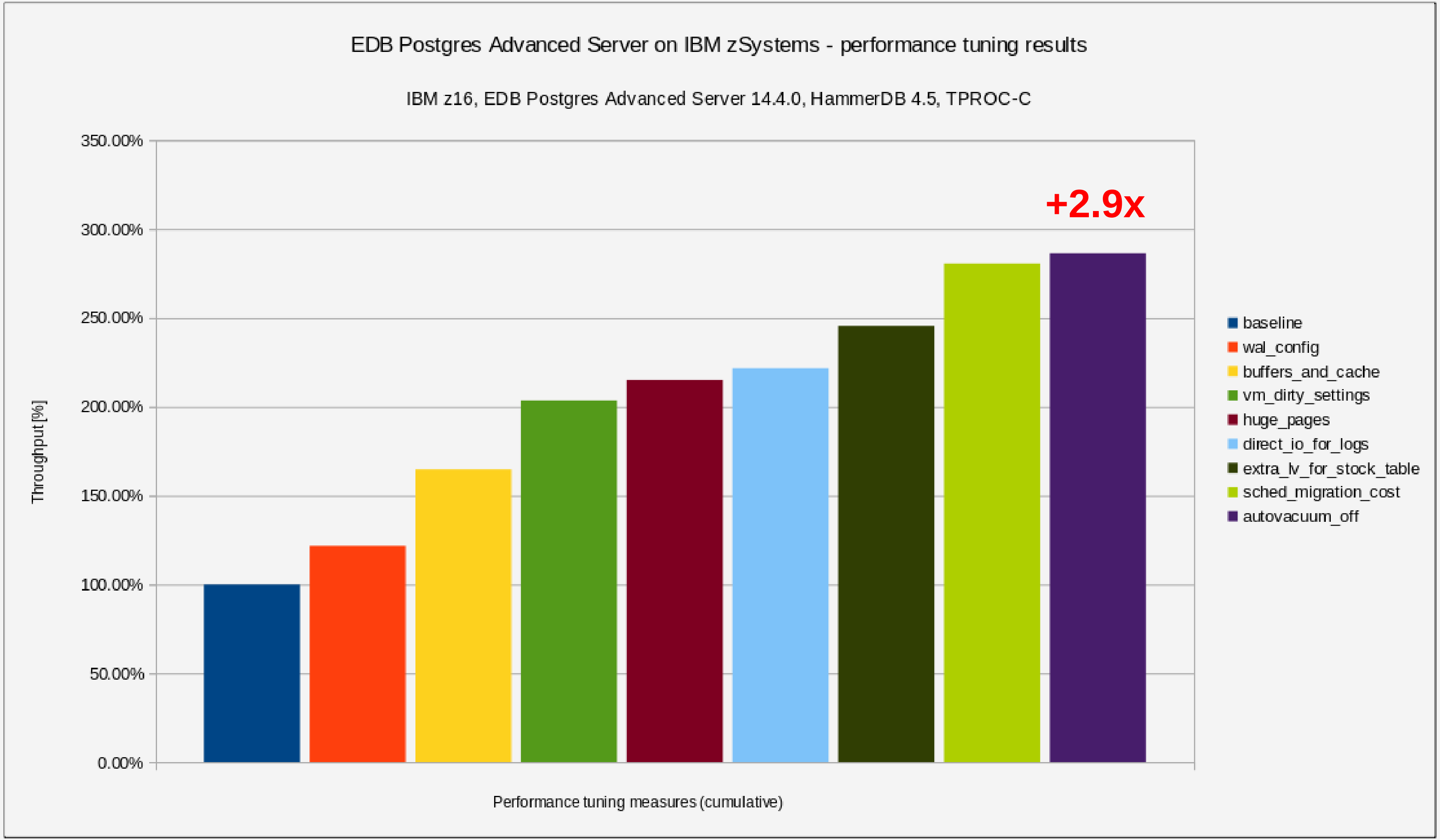
Agenda

- Introduction to EDB Postgres Advanced Server
- Performance measurement and tuning approach
- Observations and recommendations
- Summary

Disclaimers

The following is ***important*** – please read it carefully

- The performance test results in the following charts were obtained in a ***controlled lab environment*** natively in a *Logical Partition* (LPAR). The measured differences in throughput might not be observed in real-life scenarios and environments other than native LPAR.
- All of the test runs were performed with Red Hat® Enterprise Linux 8, EDB Postgres Advanced Server 14.4.0, and HammerDB 4.5. ***Other*** product versions might produce ***different*** performance results.
- All of the tests were specifically executed for ***EPAS***. The impact of the recommendations in this chart deck on ***other*** database management systems might be ***totally different***, including ***adverse*** performance effects.
- All of the tests were specifically executed for a heavy ***Online Transaction Processing*** (OLTP) workload. The impact of the recommendations in this chart deck on other types of workloads – *Online Analytical Processing* (OLAP), for example – might be ***totally different***, including ***adverse*** performance effects.



EDB Postgres Advanced Server on IBM zSystems - performance tuning results
IBM z16, EDB Postgres Advanced Server 14.4.0, HammerDB 4.5, TPROC-C
+2.9x
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
Throughput [%]
Performance tuning measures (cumulative)
baseline
wal_config
buffers_and_cache
vm_dirty_settings
huge_pages
direct_io_for_logs
extra_lv_for_stock_table
sched_migration_cost
autovacuum_off

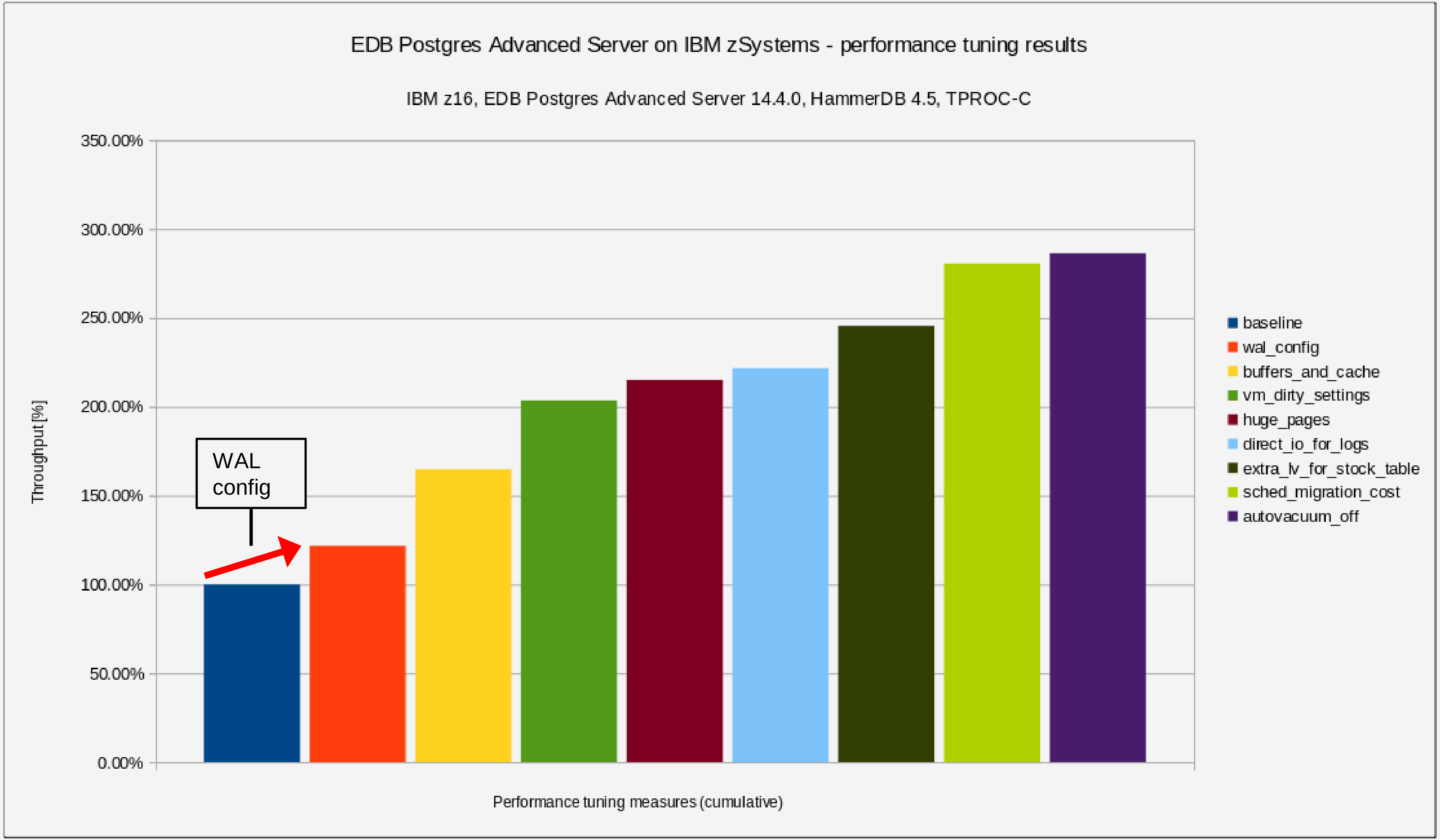
Recommendation #1

- Recommendation #1: adjust your ***Write Ahead Log*** (WAL) configuration
- Background: transactions are first written to the WAL before they are persisted to the actual database manager ***data files*** in a so-called ***checkpoint*** operation
- EPAS log is ***pretty clear*** on what happens and what should be done about it:

```
2023-03-16 11:32:44 EDT LOG:  checkpoint starting: wal

2023-03-16 11:32:54 EDT LOG:  checkpoint complete: wrote 483625 buffers (46.1%); 0 WAL file(s)
added, 596 removed, 0 recycled; write=5.702 s, sync=2.924 s, total=9.524 s; sync>

2023-03-16 11:32:54 EDT LOG:  checkpoints are occurring too frequently (10 seconds apart)

2023-03-16 11:32:54 EDT HINT:  Consider increasing the configuration parameter "max_wal_size".
```

Recommendation #1, *continued*

- In order to change this setting, you have to edit the `postgresql.conf` configuration file
  - Setting is called `max_wal_size` and I set it to `256GB` in my test environment
- In my previous PostgreSQL tuning guide, I set this value to `16GB`, but since (a) the IFLs in my current test environment are ***much more powerful*** compared to 2017 and (b) the storage server is ***NVMe-based*** compared to the formerly used spinning disks, I had to use a ***much higher*** value for this study
  - Actual "best fit" value in your environment might be different
- After applying this value and restarting EPAS, the WAL warnings / hints were gone
- ***Downside***: larger / more WAL files lead to ***longer recovery times*** in case of a failure


EDB Postgres Advanced Server on IBM zSystems - performance tuning results
IBM z16, EDB Postgres Advanced Server 14.4.0, HammerDB 4.5, TPROC-C
Throughput [%]
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
WAL config
Performance tuning measures (cumulative)
baseline
wal_config
buffers_and_cache
vm_dirty_settings
huge_pages
direct_io_for_logs
extra_lv_for_stock_table
sched_migration_cost
autovacuum_off

Recommendation #2

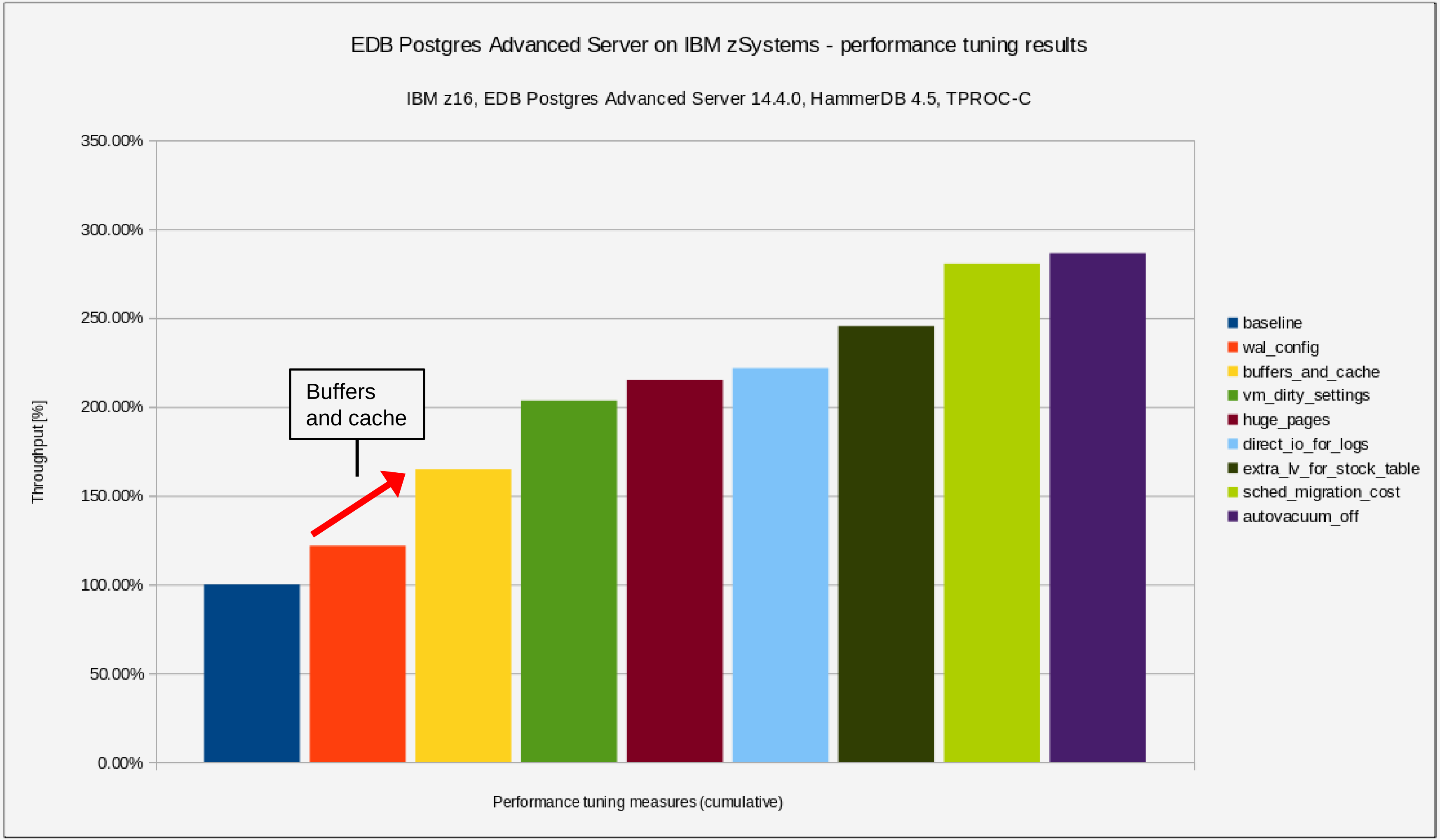
- Recommendation #2: increase EPAS' ***buffers and caches***
- Basically, this recommendation is about 2 different settings in the `postgresql.conf` configuration file:
  - `effective_cache_size`
  - `shared_buffers`
- One of the major tuning knobs: ***+35% more throughput*** in my series of test runs
- Background: the ***effective cache size*** setting is not an actual memory allocation, but *"… an **estimate** of how much memory is available for disk caching by the operating system and within the database itself"* (quote from the PostgreSQL [Tuning Wiki])
  - Used by the PostgreSQL query planner

Recommendation #2, *continued*

- The `shared_buffers` configuration parameter, on the other hand, *"…determines how much memory is dedicated to PostgreSQL to use for* ***caching data****"* (Tuning Wiki quote)
  - Unlike `effective_cache_size`, this is a ***real*** cache
- Tuning Wiki recommendation is to set `shared_buffers` to ***¼ of your Linux image's memory*** and `effective_cache_size` to ¾ of the memory
  - This is what I used in the 2017 study
- Since my test environment had 128 GiB of memory available, my configuration looked like this:
  - `effective_cache_size = 96GB`
  - `shared_buffers = 32GB`

Recommendation #2, *continued*

- In the 2017 study, configuring the `shared_buffers` value too high led to ***PostgreSQL terminations*** ("abends") caused by the Linux *Out-Of-Memory* (OOM) killer process
  - Could ***not*** reproduce those abends in the current study – not even for very high `shared_buffers` values
- In the 2017 study, increasing `shared_buffers` ***beyond*** ¼ of the memory did ***not*** increase end-to-end throughput any further
  - However, in the current study, increasing `shared_buffers` to ½ of the available memory ***did increase*** the overall end-to-end throughput ***by an additional ca. 14%*** (on top of the mentioned +35%)
  - In the end, I decided to stick with ¼ of the memory for `shared_buffers`, in order to stay in line with the official Tuning Wiki recommendation


EDB Postgres Advanced Server on IBM zSystems - performance tuning results
IBM z16, EDB Postgres Advanced Server 14.4.0, HammerDB 4.5, TPROC-C
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
Throughput [%]
Buffers and cache
Performance tuning measures (cumulative)
baseline
wal_config
buffers_and_cache
vm_dirty_settings
huge_pages
direct_io_for_logs
extra_lv_for_stock_table
sched_migration_cost
autovacuum_off

Recommendation #3

- Recommendation #3: configure the writeback of ***dirty memory*** (i.e., dirty pages) in Linux

- This task consists of changing the values of 2 related settings:
  - `vm.dirty_background_bytes` (counterpart setting in %: `vm.dirty_background_ratio`)
  - `vm.dirty_bytes` (counterpart setting in %: `vm.dirty_ratio`)

- Background – see Linux kernel documentation:
  - The `dirty_background_bytes` setting *"contains the amount of dirty memory at which the* ***background kernel flusher threads*** *will start writeback"* (quote from the above link)
  - The `dirty_bytes` setting, on the other hand, *"contains the amount of dirty memory at which a* ***process generating disk writes will itself*** *start writeback"* (quote)

Recommendation #3, *continued*

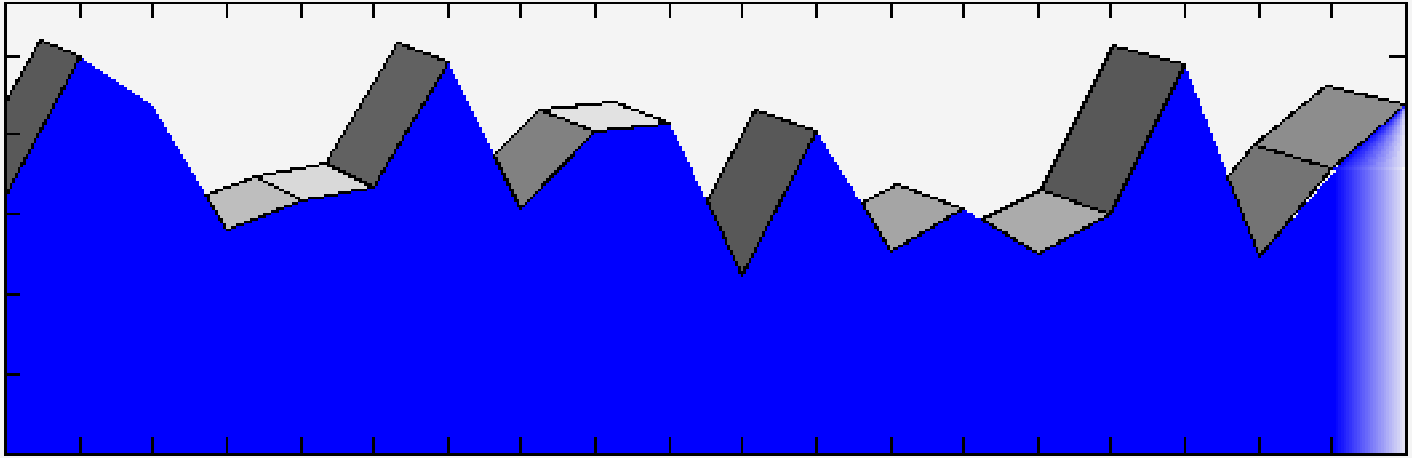
- The ***default*** values for those 2 settings are (a) configured in ***percent*** and (b) result in ***pretty large values*** for Linux images with large amounts of memory
  - For example, `vm.dirty_background_ratio` resulted in 12.8 GiB in my test environment (default value of this setting is 10%, and 10% of 128 GiB is 12.8 GiB)
  - For both settings, ***only 1 variant*** can be active at any given point in time: either the value in percent, or the absolute value in bytes

- Recommendation:
  - `vm.dirty_background_bytes=67108864` (64 MiB)
  - `vm.dirty_bytes=536870912` (512 MiB)

Recommendation #3, *continued*

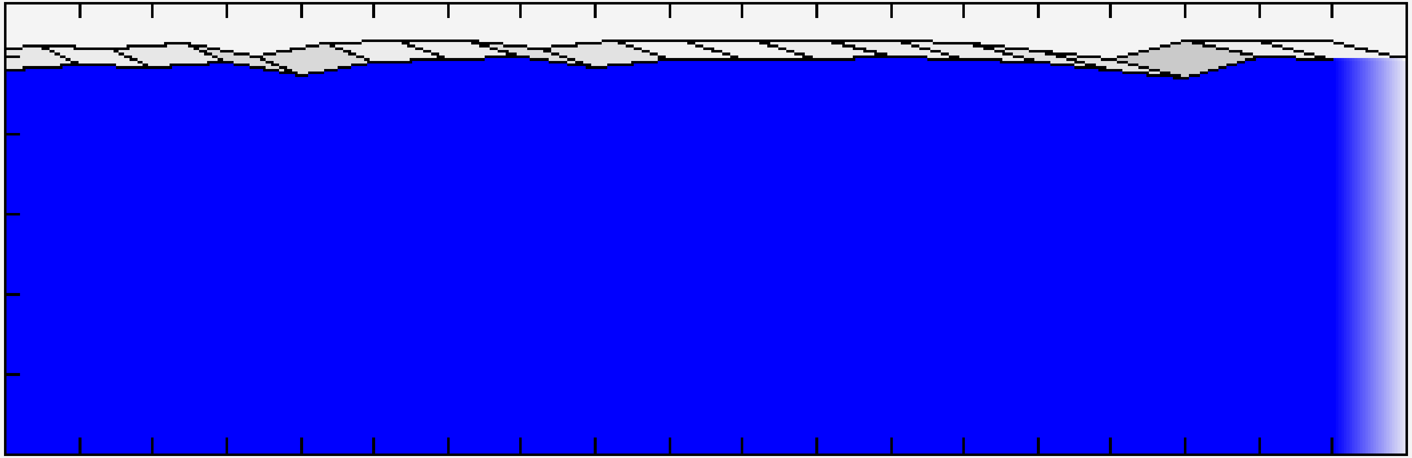
- This will force the writeback of dirty memory ***much earlier*** compared to the default settings, which in turn results in a ***much more balanced*** disk I/O behavior and ***much smoother*** end user response times
  - This recommendation avoids ***burst waves*** in disk I/O
  - Therefore, the mentioned settings are more about ***response time*** than throughput
- In order to ***persist*** those values across reboots, add them to the `/etc/sysctl.conf` configuration file and either (a) reboot your Linux image or (b) run the `sysctl -p` command

Throughput graph(*) ***before*** configuration changes

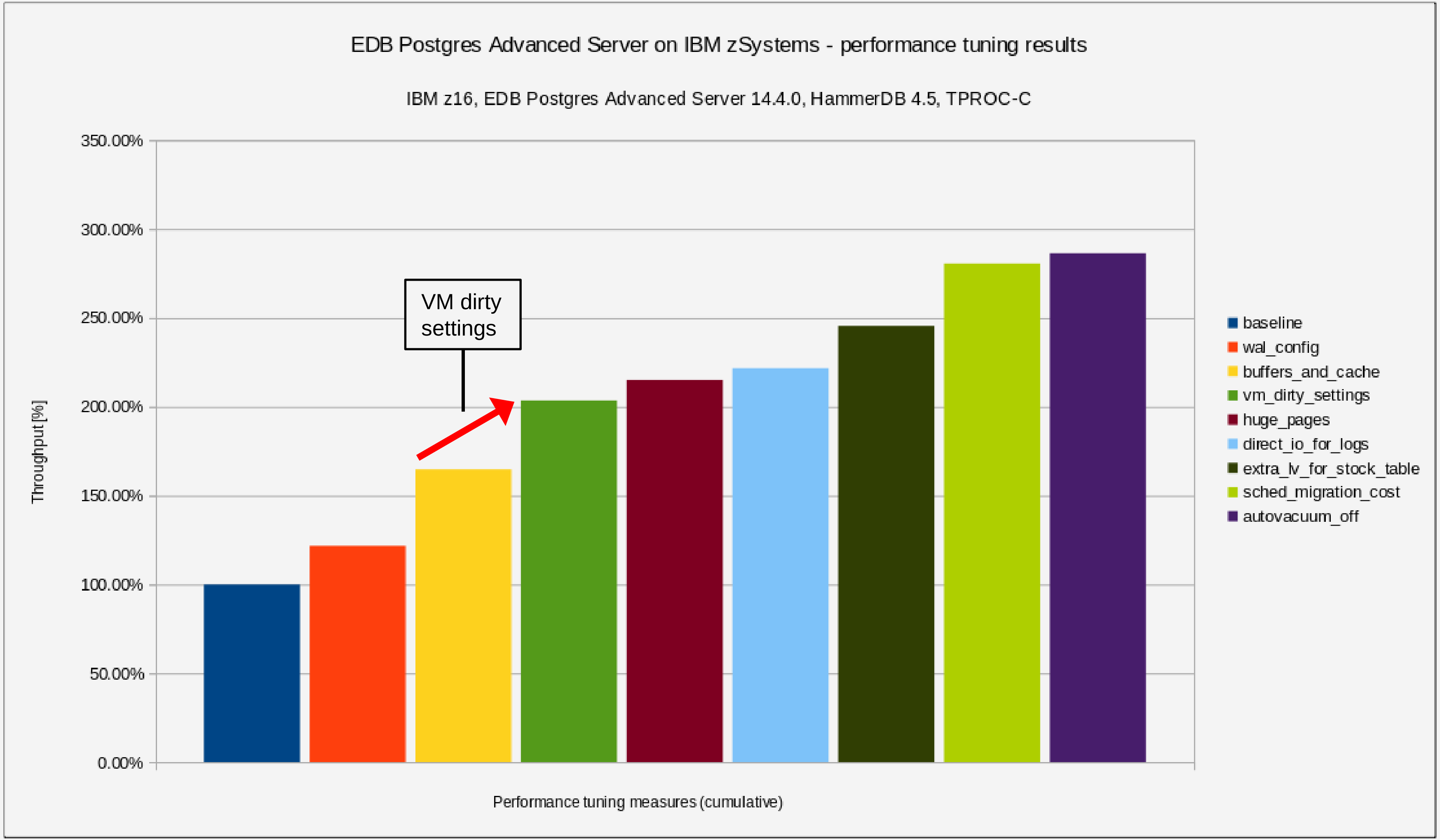


(*)**Note**: Of course, response time is not equal to throughput. However, from the throughput graph, one can conclude that the response times must have been very shaky, because the amount of virtual users didn't change after the ramp-up phase.

Throughput graph(*) *after* configuration changes



(*)**Note**: A stable overall throughput graph is not an absolute guarantee that the response times of the individual virtual users were 100% consistent during the entire test run. It is, however, a very strong indicator.


EDB Postgres Advanced Server on IBM zSystems - performance tuning results
IBM z16, EDB Postgres Advanced Server 14.4.0, HammerDB 4.5, TPROC-C
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
Throughput [%]
VM dirty settings
Performance tuning measures (cumulative)
baseline
wal_config
buffers_and_cache
vm_dirty_settings
huge_pages
direct_io_for_logs
extra_lv_for_stock_table
sched_migration_cost
autovacuum_off

Recommendation #4

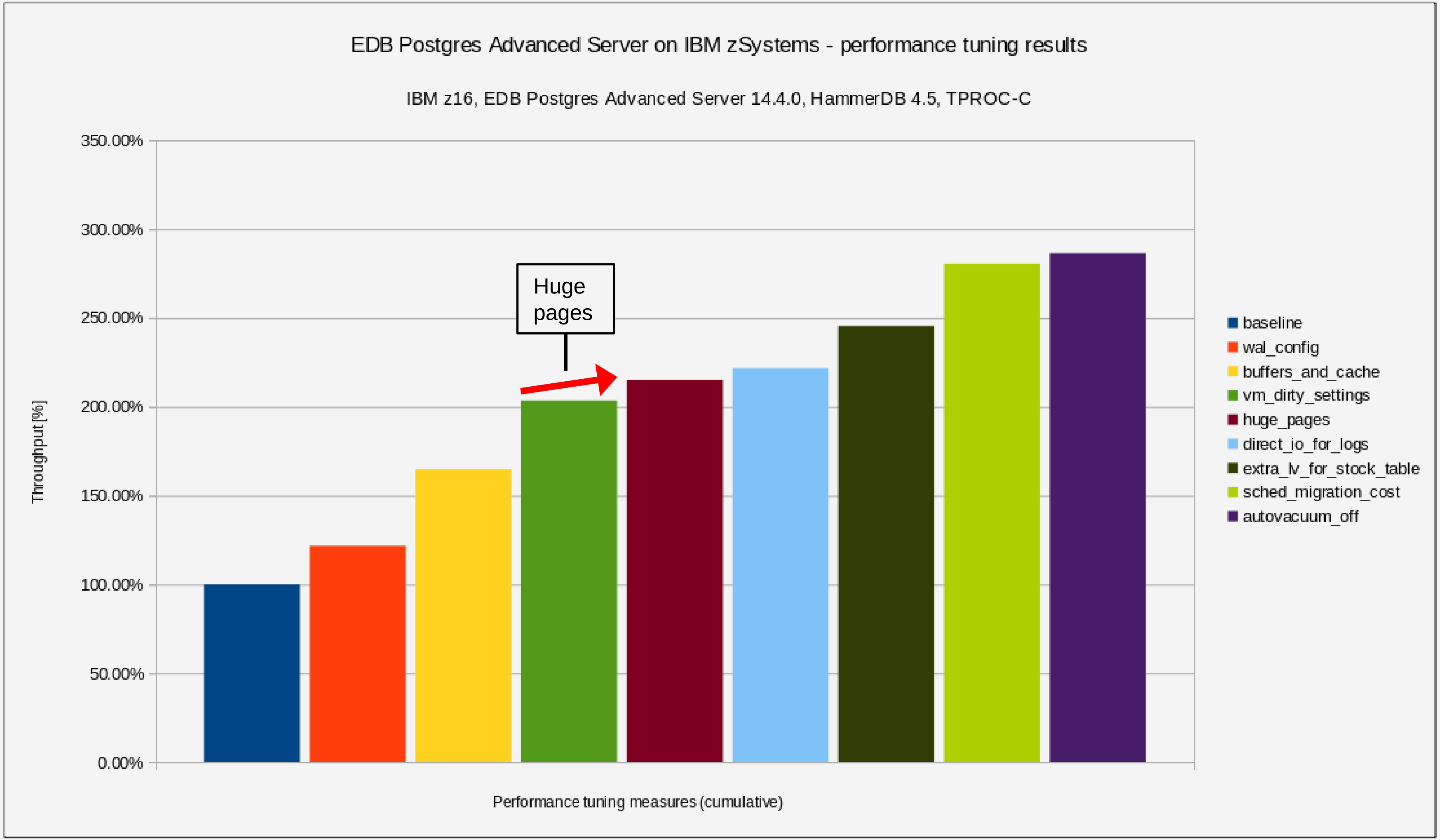
- Recommendation #4: configure ***huge pages*** for your Linux image

- Huge pages in this context means ***persistent*** huge pages, which are controlled via the `vm.nr_hugepages` kernel parameter – see also the corresponding Linux kernel [documentation] regarding huge pages
  - Independent from configuring persistent huge pages, it is also strongly recommended to ***turn off transparent*** huge pages for databases (see [this link] for a good summary of the problem)

- The configured amount of huge pages in the Linux image must be ***greater than*** the value of the `shared_buffers` setting – EPAS requires some additional memory for other shared data
  - When the `huge_pages` parameter is set to `on` in `postgresql.conf`, shared buffers ***must fit*** into the configured amount of huge pages in your Linux image, otherwise EPAS won't start up

Recommendation #4, *continued*

- In order to ***determine*** the actual amount of huge pages that is required for your environment, start EPAS without huge pages enabled and follow the ***HOWTO*** in the PostgreSQL documentation
- In order to ***persist*** the huge pages configuration across reboots, add it to `/etc/sysctl.conf`
  - `vm.nr_hugepages=38912` (38 GiB total = 32 GiB for `shared_buffers` + 6 GiB other shared data)
- With ***smaller*** `shared_buffers` values, huge pages do ***not*** have a huge impact on performance
  - The impact in my series of test runs was ca. 5.7%, see the bar chart on the next slide
- With ***larger*** `shared_buffers` values, the impact of huge pages is ***noticeably higher***
  - In my test environment, I measured ***ca. 11% higher*** end-to-end throughput when exploiting huge pages for a ***large*** `shared_buffers` configuration ("large" here means ½ of the available Linux memory)


EDB Postgres Advanced Server on IBM zSystems - performance tuning results
IBM z16, EDB Postgres Advanced Server 14.4.0, HammerDB 4.5, TPROC-C
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
Throughput [%]
Huge pages
Performance tuning measures (cumulative)
baseline
wal_config
buffers_and_cache
vm_dirty_settings
huge_pages
direct_io_for_logs
extra_lv_for_stock_table
sched_migration_cost
autovacuum_off

Recommendation #5

- Recommendation #5: move ***disk I/O heavy tables*** into their own tablespaces
- When analyzing disk I/O performance related data from my test environment, I noticed that the logical volume that held the database ***data files*** was permanently running at 100% utilization
- Output of Linux `iostat` ***before*** the change:

```
03/31/2023 04:36:45 AM
Device              r/s        w/s      rkB/s       wkB/s  r_await  w_await   aqu-sz  rareq-sz  wareq-sz  svctm   %util
...
edb_data_enc   93870.40  126297.20  923960.00  1082067.20     0.46    11.35  1477.52      9.84      8.57   0.00  100.00
edb_logs_enc       0.00    2028.40       0.00   345965.60     0.00     0.39     0.80      0.00    170.56   0.49  100.00
```

- **Note**: some columns were cut in the above output in order to make it fit the page

Recommendation #5, *continued*

- By putting the ***tables that grew the most*** during a benchmark run into their own tablespaces one at a time, I figured out that the `STOCK` table made up for ***more than 50%*** of the overall disk I/O traffic

- Created a ***separate table space*** for the `STOCK` table:
  - `CREATE TABLESPACE STOCK_TS LOCATION '/edb_stock';`
  - `ALTER TABLE STOCK SET TABLESPACE STOCK_TS;`

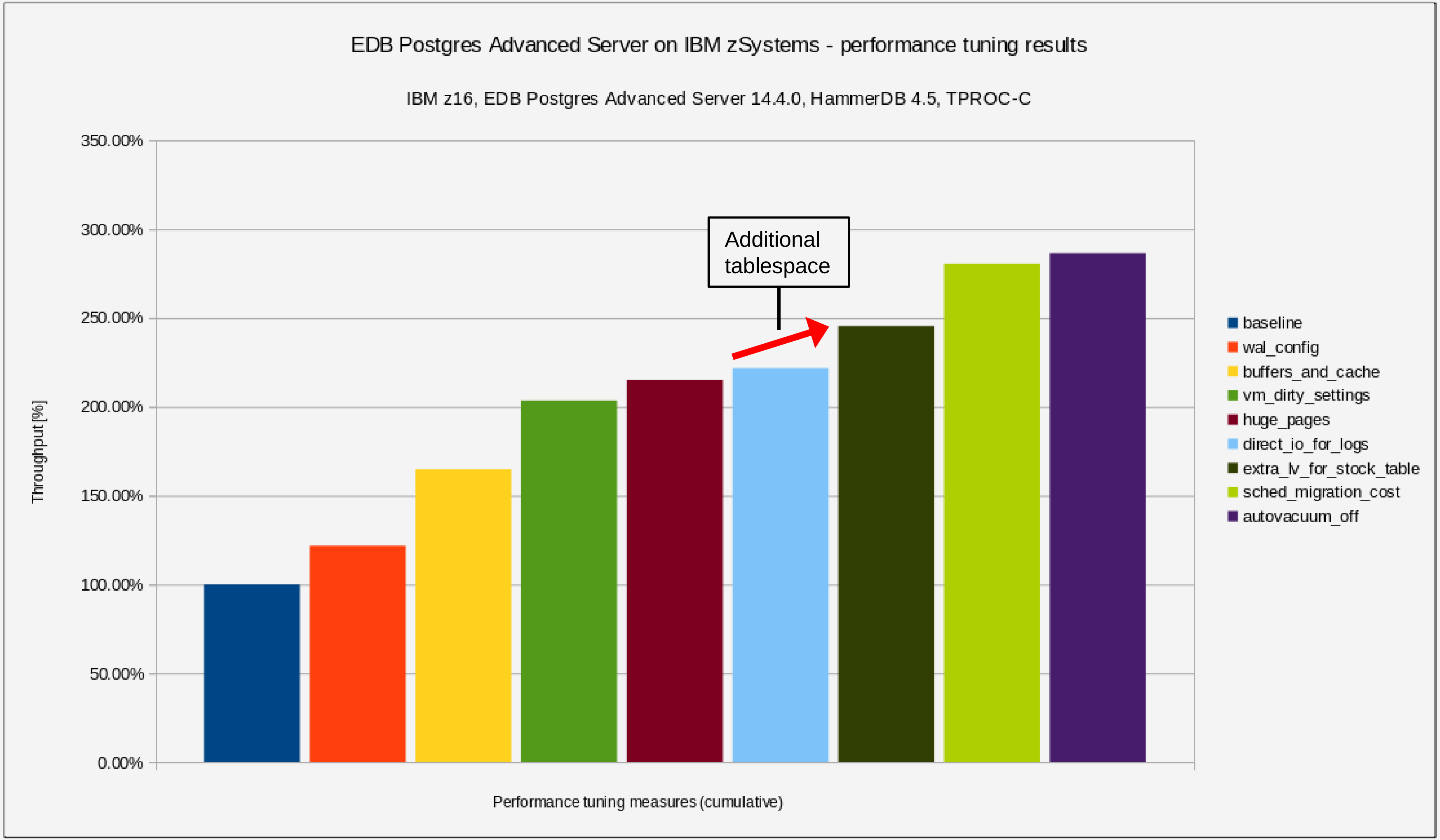
- As can be seen above, the `STOCK` table got its ***own mount point***, which was realized using an additional ***striped logical volume***, which in turn was based on its own ***separate set*** of SCSI disks
  - This resulted in overall higher disk I/O bandwidth

Recommendation #5, *continued*

- Linux `iostat` output ***after*** the change:

```
03/31/2023 05:17:29 AM
Device                r/s       w/s       rkB/s       wkB/s  r_await  w_await  aqu-sz  rareq-sz  wareq-sz  svctm   %util
...
edb_data_enc     36218.00  38604.60   358990.40   374364.80     0.44     6.44  264.49      9.91      9.70   0.01  100.00
edb_stock_enc    63277.20  89612.80   599930.40   719316.80     0.33     7.32  676.84      9.48      8.03   0.01  100.00
edb_logs_enc         0.00   2110.20        0.00   314109.60     0.00     0.37    0.78      0.00    148.85   0.47  100.00
```

- **Note**: some columns were cut in the above output in order to make it fit the page



EDB Postgres Advanced Server on IBM zSystems - performance tuning results
IBM z16, EDB Postgres Advanced Server 14.4.0, HammerDB 4.5, TPROC-C
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
Throughput [%]
Additional tablespace
Performance tuning measures (cumulative)
baseline
wal_config
buffers_and_cache
vm_dirty_settings
huge_pages
direct_io_for_logs
extra_lv_for_stock_table
sched_migration_cost
autovacuum_off

Recommendation #6

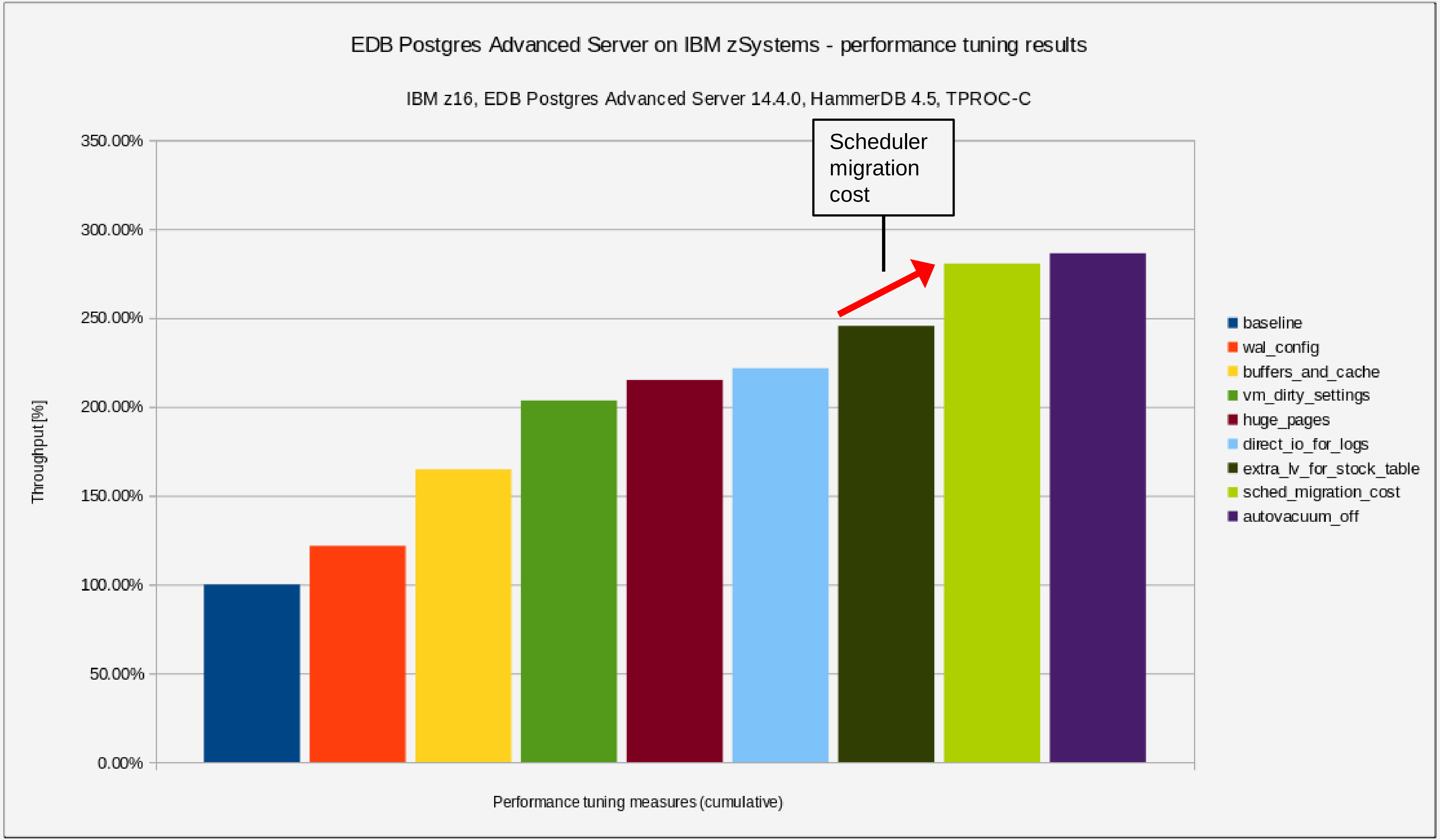
- Recommendation #6: lower the Linux kernel ***scheduler migration cost*** setting

- Discovered this tunable as part of my 2017 study
  - Over time, this setting has proven to improve throughput for a ***large number*** of different workloads
  - Chances are high that this setting is going to help if you (a) have a ***large amount*** of active tasks in the system, (b) ***idle time*** in `top` / `sar` / `vmstat`, (c) ***no obvious bottleneck*** like disk I/O or networking, and (d) ***no obvious locking*** inside the application

- The setting is called `kernel.sched_migration_cost_ns` and it configures the number of nanoseconds the kernel will wait ***before considering moving a task*** to another CPU
  - The ***higher*** this migration cost is, the ***longer*** the kernel scheduler will wait before considering moving a task to another CPU

Recommendation #6, *continued*

- My recommendation is to ***lower*** the value of this parameter by ***one order of magnitude***
  - Found the optimal setting empirically by performing test runs with a large series of values
- In my test environment, this ***increased*** end-to-end throughput ***by ca. 14%***
- In order to ***persist*** this configuration change, add the following line to `/etc/sysctl.conf`:
  - `kernel.sched_migration_cost_ns=50000`
- Besides the increased throughput, another ***indication*** that the Linux image is performing ***more useful work*** with this setting applied is that the amount of ***user time*** (%usr) increases measurably
  - Can be verified with `top` and / or `sar` data



EDB Postgres Advanced Server on IBM zSystems - performance tuning results
IBM z16, EDB Postgres Advanced Server 14.4.0, HammerDB 4.5, TPROC-C
Scheduler migration cost
Throughput [%]
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
Performance tuning measures (cumulative)
baseline
wal_config
buffers_and_cache
vm_dirty_settings
huge_pages
direct_io_for_logs
extra_lv_for_stock_table
sched_migration_cost
autovacuum_off

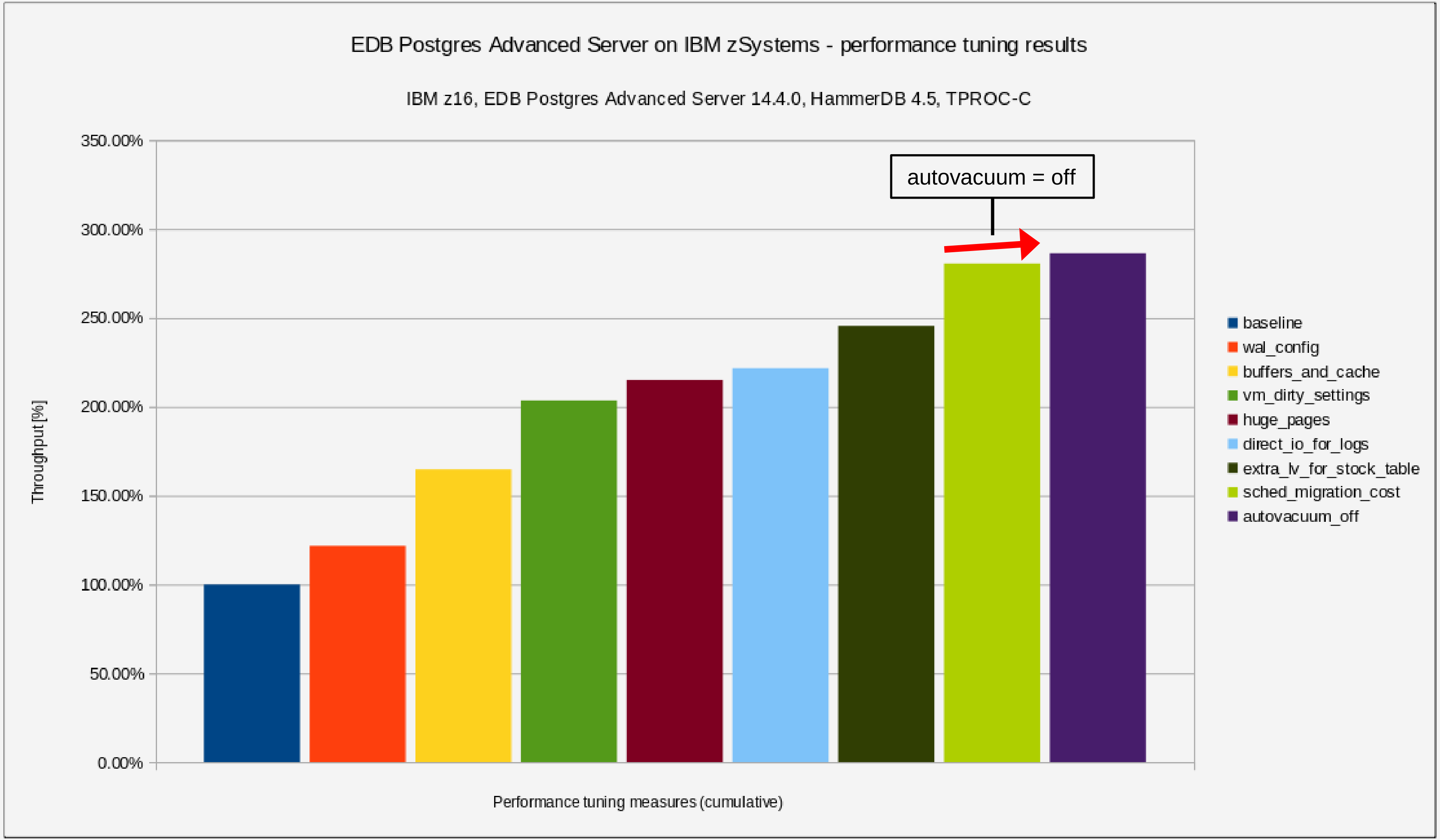
Recommendation #7

- Recommendation #7: for ***mostly read-only*** workloads, consider turning off the ***autovacuum daemon***

- In general, it is advisable to ***keep*** the autovacuum daemon ***turned on***
  - The autovacuum daemon is turned on ***by default***

- The autovacuum daemon does ***very useful things***
  - Recovers or reuses disk space occupied by updated or deleted rows
  - Updates ***statistics*** used by the PostgreSQL query planner

- However, this process does cost CPU cycles – impact of turning autovacuum off for the HammerDB TPROC-C workload: ***ca. 2% more throughput*** (see bar chart)

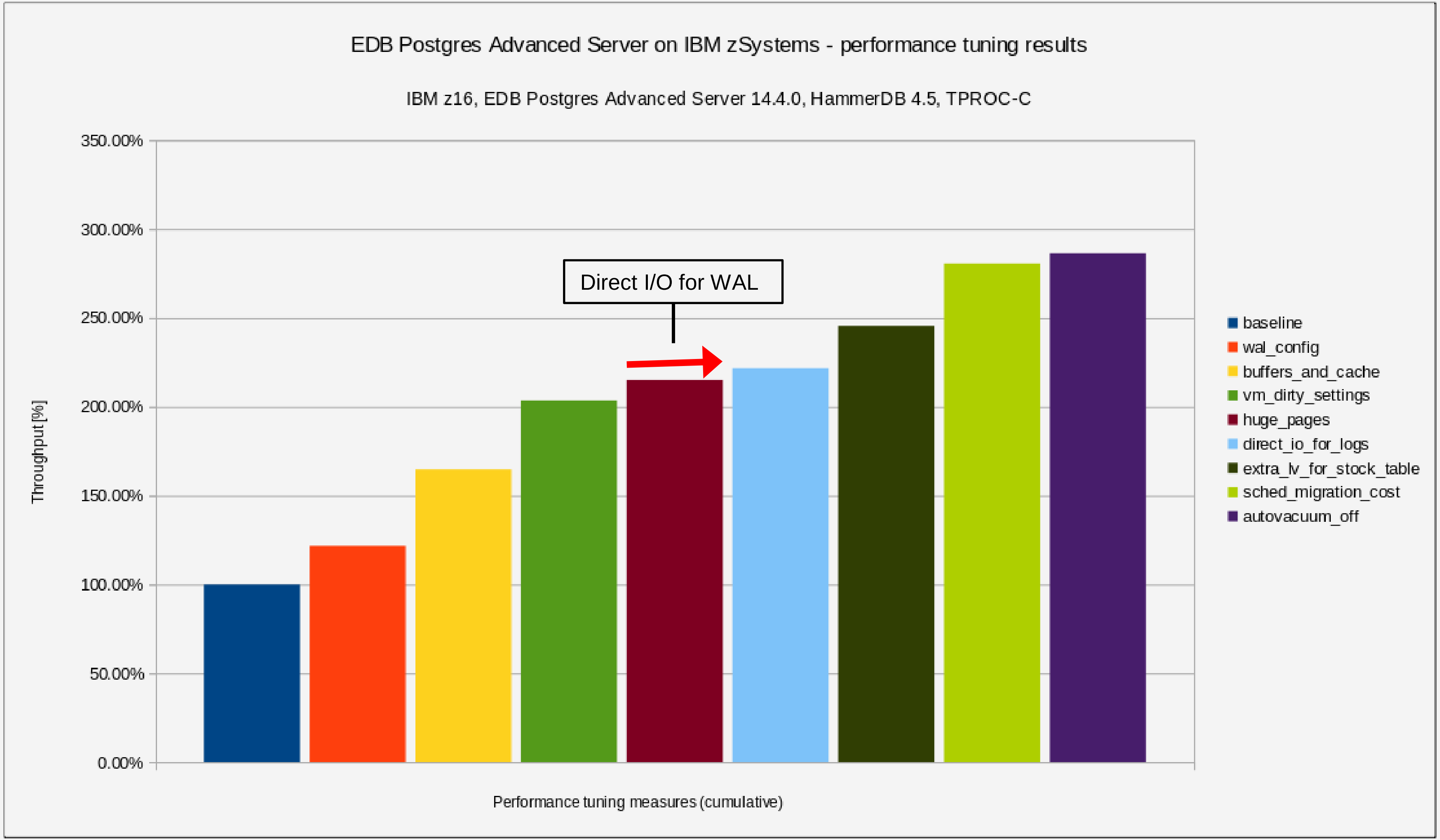

EDB Postgres Advanced Server on IBM zSystems - performance tuning results
IBM z16, EDB Postgres Advanced Server 14.4.0, HammerDB 4.5, TPROC-C
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
Throughput [%]
autovacuum = off
Performance tuning measures (cumulative)
baseline
wal_config
buffers_and_cache
vm_dirty_settings
huge_pages
direct_io_for_logs
extra_lv_for_stock_table
sched_migration_cost
autovacuum_off

Direct I/O for transaction log files

- Some people prefer to make use of ***Direct I/O*** for database transaction log files – this means ***WAL*** for EPAS
- Background: when using Direct I/O, the database management system ***bypasses*** the Linux ***page cache*** and writes change records directly to the disks, thus avoiding loss of transactions in case of a failure
- However, using Direct I/O for files only means that the data safely arrived at the ***built-in hardware cache*** of the storage server / hard drive, it doesn't mean that the data has actually been written to disk
  - This is guaranteed by the `fsync()` system call in Linux
- When opening WAL files with the `open_sync` method for the `wal_sync_method` parameter, EPAS uses ***both*** `O_DIRECT` and `O_SYNC` for the transaction log files
  - With `O_SYNC`, every `write()` system call behaves like it's followed by a call to `fsync()`

Direct I/O for transaction log files, *continued*

- I personally think that Direct I/O is ***not strictly required*** for WAL, because EPAS issues an `fsync()` anyway for transaction log entries in its default setup
  - Thus, transaction log entries are ***guaranteed*** to be written to disk, even in the default setup
- If you'd like to exploit Direct I/O for WAL anyhow and you ask yourself if this comes with a ***performance impact*** or not, the answer is "no" – see the performance bar chart on the next slide



EDB Postgres Advanced Server on IBM zSystems - performance tuning results
IBM z16, EDB Postgres Advanced Server 14.4.0, HammerDB 4.5, TPROC-C
Direct I/O for WAL
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
Throughput [%]
Performance tuning measures (cumulative)
baseline
wal_config
buffers_and_cache
vm_dirty_settings
huge_pages
direct_io_for_logs
extra_lv_for_stock_table
sched_migration_cost
autovacuum_off

Agenda

- Introduction to EDB Postgres Advanced Server
- Performance measurement and tuning approach
- Observations and recommendations
- Summary

Summary

- EDB Postgres Advanced Server ***improves*** the already very feature-rich Open Source PostgreSQL database management system even further
  - Enhanced ***Oracle compatibility*** features
- If you follow the recommendations in this tuning guide, you can ***greatly improve performance*** of your EPAS installation on Linux on IBM zSystems
  - Examples: shared buffers, huge pages, scheduler migration cost
  - ***Saturating*** all 24 IFLs of my test environment was only possible after applying all of my tuning recommendations
- ***Remember***: if you're interested in a competitive comparison against Intel x86, please consult the official IBM z16 Proof Points chart deck

Thank you!

Resources

- Linux on IBM zSystems and IBM® LinuxONE
  - Official homepage: https://www.ibm.com/z/linux
  - Documentation: https://www.ibm.com/docs/en/linux-on-systems?topic=linuxone-library-overview
  - Tuning hints and tips: https://www.ibm.com/docs/en/linux-on-systems?topic=performance-tuning-hints-tips
- EDB Postgres Advanced Server: https://www.enterprisedb.com/products/edb-postgres-advanced-server
- EDB Postgres Advanced Server documentation: https://www.enterprisedb.com/docs/epas/latest
- EDB Enterprise Postgres with IBM 14.4 adds support for Linux on IBM Z® and IBM® LinuxONE: https://www.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_ca/3/897/ENUS222-173/index.html
- PostgreSQL homepage: https://www.postgresql.org
- PostgreSQL documentation: https://www.postgresql.org/docs
- PostgreSQL Tuning Wiki: https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server
- Tutorial on developer.ibm.com: https://developer.ibm.com/tutorials/postgresql-experiences-tuning-recomendations-linux-on-ibm-z