Accelerating PostgreSQL Backups with zlib-accel

Enhancing database backup speed with advanced compression offload

intel

Presenting on behalf of my colleague

- Matt Welch, Java Platform Engineering
- Open Source contributor

About Matt!

Zlib-accel Foundations and Acceleration Architecture

Understanding zlib-accel

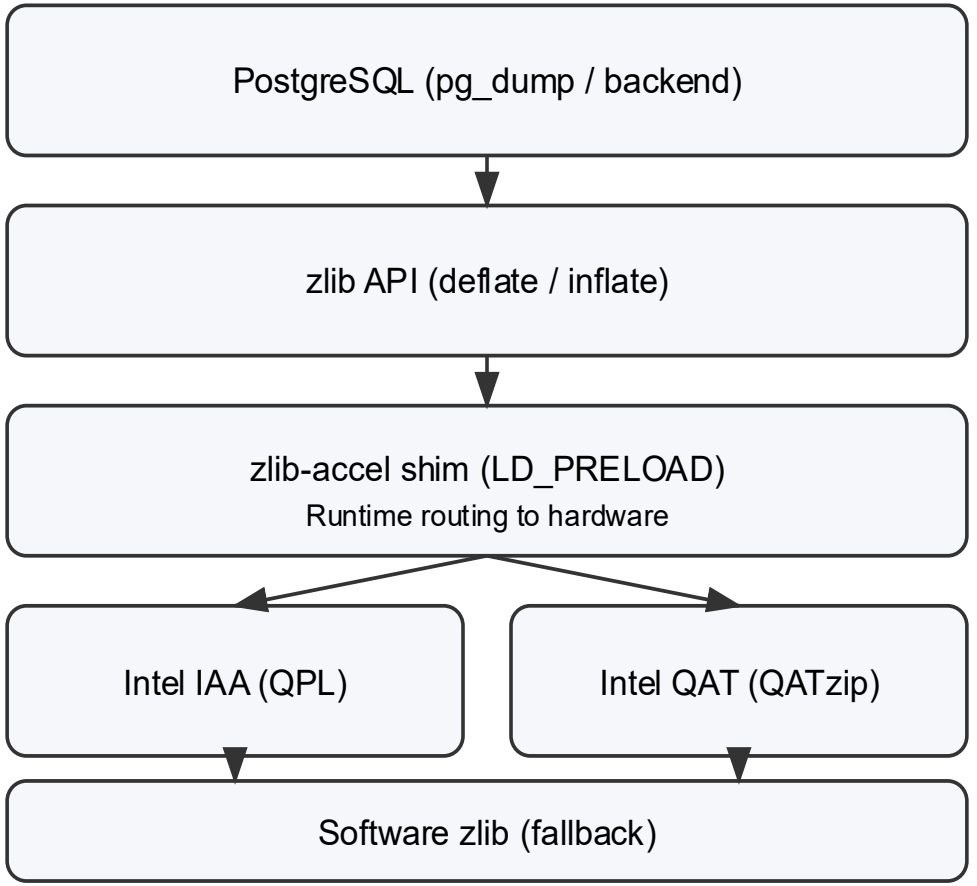
- Transparent compression acceleration layer
- Intercepts standard zlib (inflate/deflate) calls
- Offloads compression to Intel® Xeon® hardware accelerators:
  - Intel® QuickAssist Technology (QAT)
  - Intel® In-Memory Analytics Accelerator (IAA)
- No application or workflow changes
- Safe fallback to software zlib
- Validated on Linux

github.com/intel/zlib-accel

How zlib-accel Interfaces with Intel Hardware

- Intercepts zlib calls at runtime with LD_PRELOAD
- Determines hardware compatibility per request
- Routes Deflate to Intel QAT or IAA
- Manages queues, buffers, and contexts
- Falls back to software

Control/Dataflow Graph of Software Stack with Interfaces



PostgreSQL Integration and Operational Enhancements

PostgreSQL Backup Workload and Test Setup

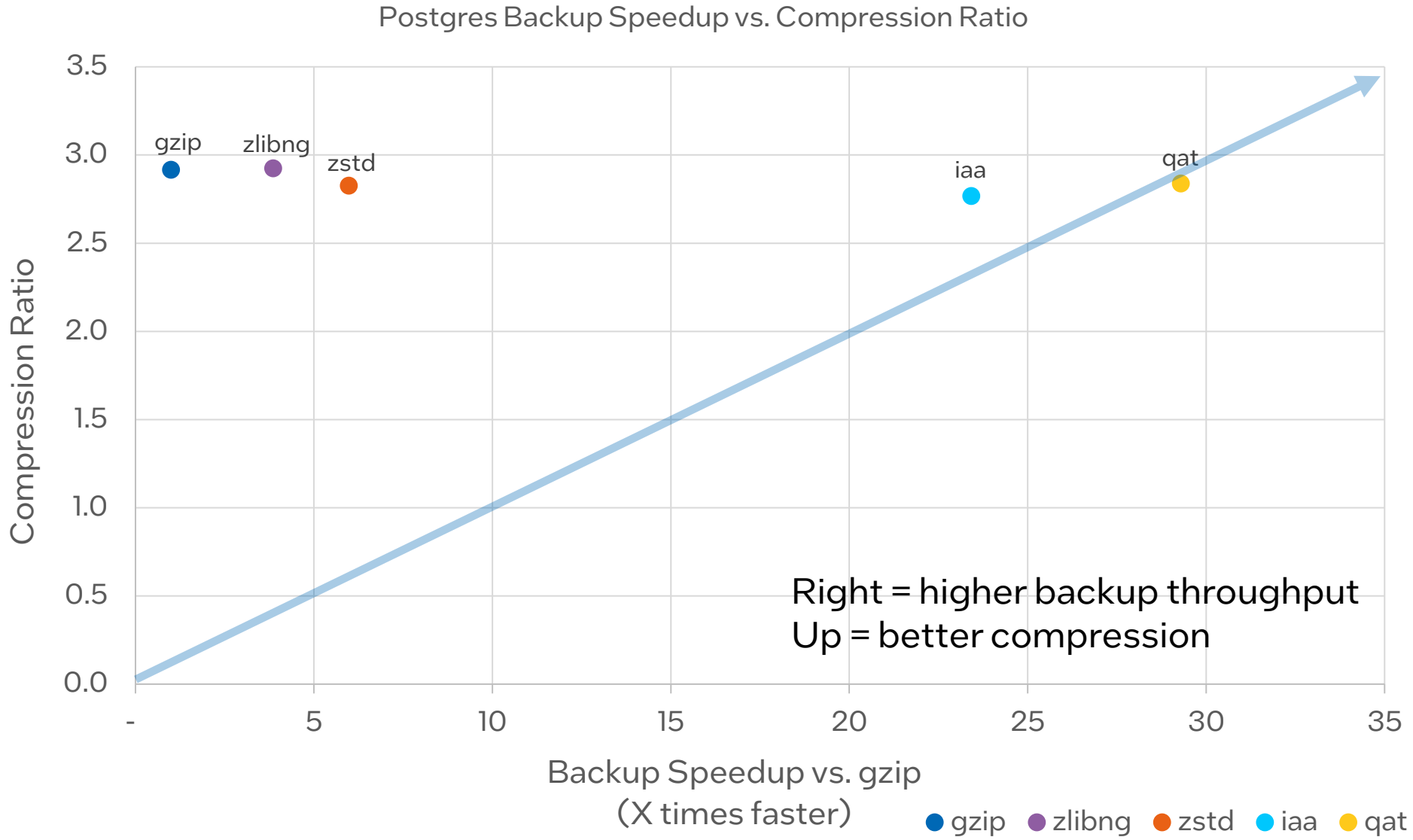
- pg_dump backup workload with compression
- Compression throughput dominates backup time
- Four parallel workers using pg_dump jobs
- PostgreSQL 17.4 on AWS i7i.**metal**-24xl
- Compared zlib, zlib-ng, zstd, zip, QAT and IAA

Sample Invocation

```
export LD_PRELOAD=<path_to_shim>

pg_dump <db_name> --create --
format=directory --compress=gzip
<other_options>

pg_restore --clean --create
<other_options>

unset LD_PRELOAD
```

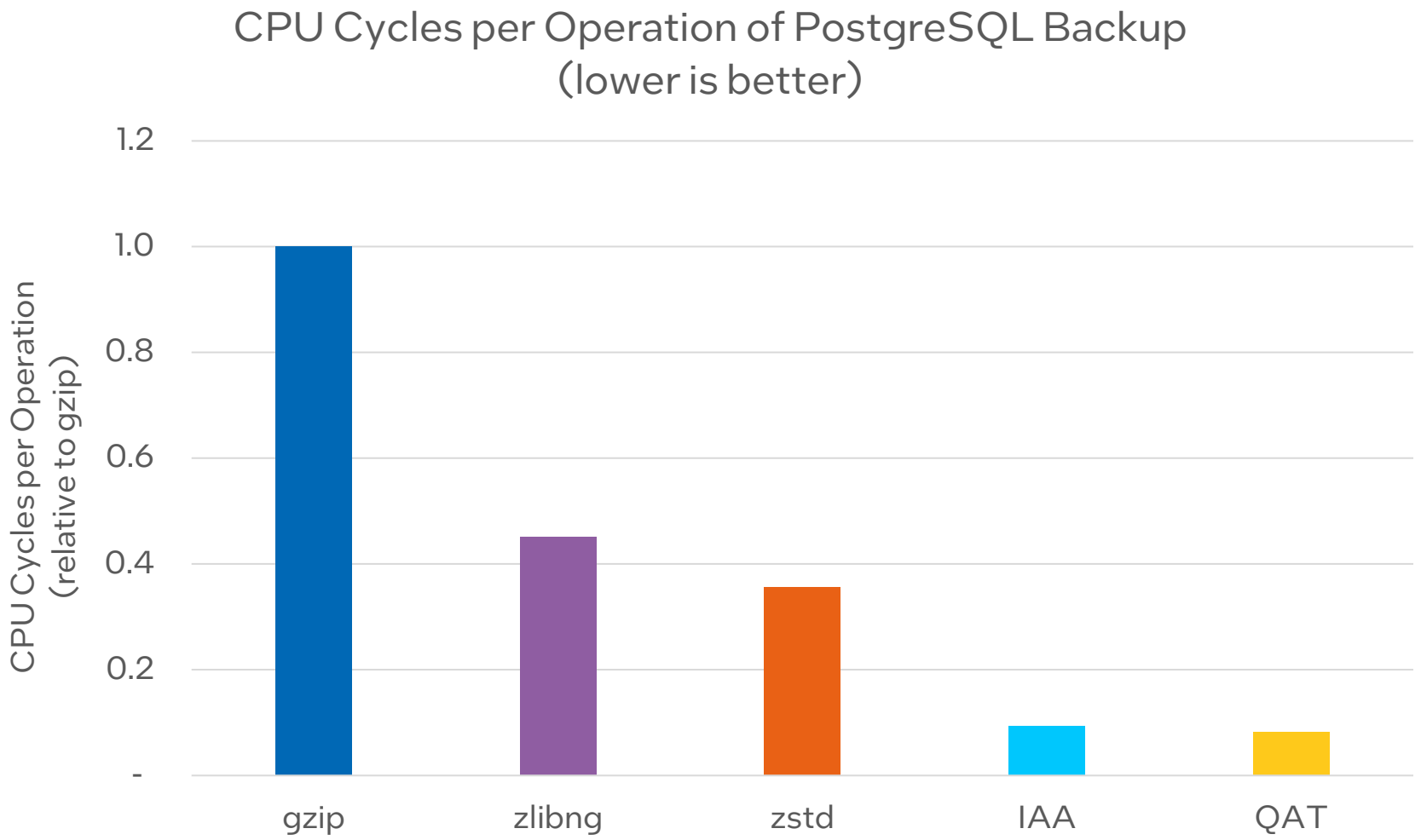
Performance Summary

PostgreSQL backup compression tradeoffs



CPU per Operation Cycles

Relative to gzip



Performance Gains vs. Software zlib

- Up to 29x higher throughput with QAT
- Up to 23x higher throughput with IAA
- Large reduction in CPU cycles per backup
- Minor compression ratio differences
- Scales well with parallel pg_dump jobs

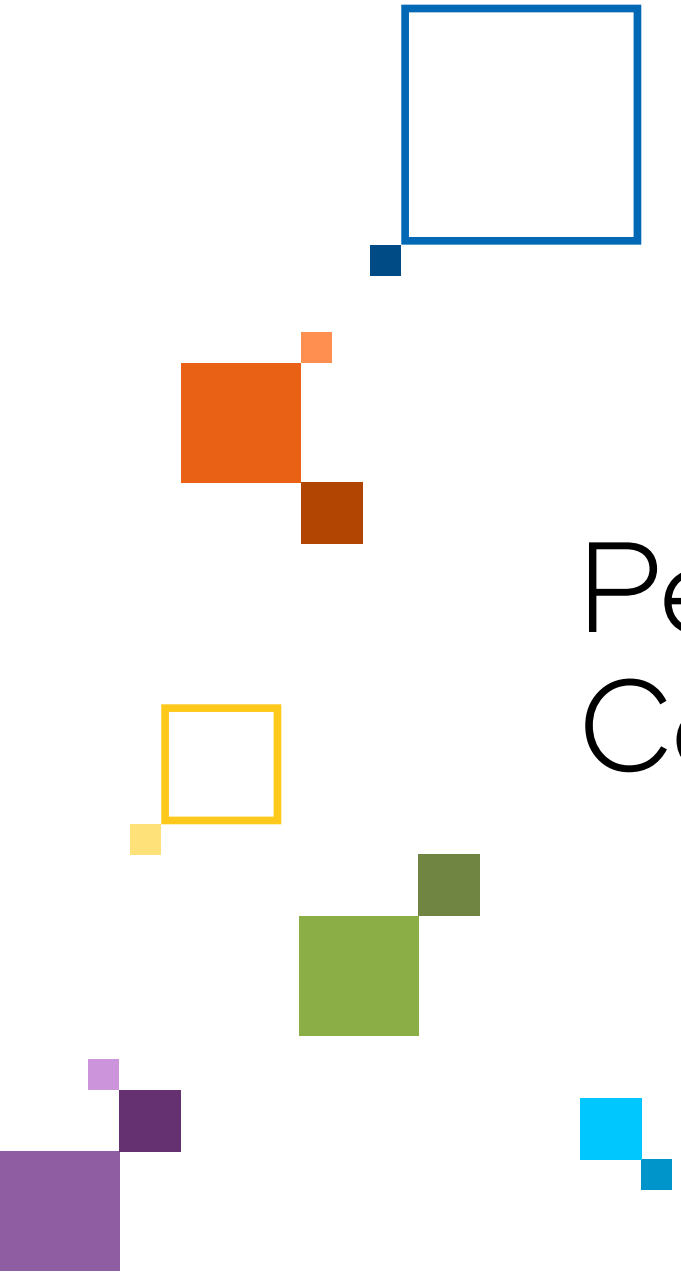
Operational Impact and Strategic Value

Operational Advantages for PostgreSQL Administrators

- Faster backup times
- Fewer CPU cycles spent on compression
- Minimal operational risk
- Better cost and resource efficiency

Strategic Benefits of Hardware-Accelerated Compression

- Scales as data volumes grow
- Integrates cleanly with existing workflows
- Speeds up downstream analytics
- Improves efficiency and sustainability



Performance Tradeoffs: Key Considerations and Takeaways

Considerations

- ✓Compression-heavy workloads benefit most
- ✓Sustained, large data streams are ideal
- ✓Modern Intel® Xeon® platforms required
- ✓Small compression tradeoff vs large speed gain
- ✓Drop-in deployment with safe fallback

Key Takeaways

- Hardware accelerated compression for PostgreSQL backups
- Up to 29x higher backup throughput over zlib
- Fewer CPU cycles spent on compression
- Drop-in deployment with no application changes
- Clear operational and strategic benefits

Acknowledgements (The Real MVPs!)

Experimental design, data collection, and analysis

- Matt Welch
  matt.welch@intel.com
- Soji Denloye
  olasoji.denloye@intel.com

Q&A

github.com/intel/zlib-accel

zlib-accel whitepaper

Backup

Measurement Methodology

- Test Environment Setup
  - Benchmarks ran on AWS EC2 i7i.metal-24xl with Ubuntu 24.04.3 and PostgreSQL 17.4
- Backup and Compression Methodology
  - PostgreSQL backups used pg_dump with four parallel jobs and gzip-compatible compression (zlib)
  - Single source database consuming 100 GB
- Compression Acceleration Techniques
  - zlib-accel library redirected compression to software or hardware accelerators like Intel QAT and IAA.
- Performance Metrics and Validation
  - Using a known database, measured backup duration, compression ratio, and CPU efficiency

Q&A — zlib-accel, QATzip, and QPL

- Does zlib-accel talk directly to hardware?
- What are QATzip and QPL?
- Why not use QATzip or QPL directly?
- What does zlib-accel add?
- Does this affect compatibility?

LD_PRELOAD Mechanism

- Linux LD_PRELOAD injects libraries at runtime
- Preloaded symbols override standard libc/zlib functions
- Enables symbol interposition without recompiling
- zlib-accel intercepts Deflate calls transparently
- Routes compression to QAT / IAA when available
- Automatic fallback to software zlib