PERCONA LIVE ONLINE MAY 12 - 13th 2021

PostgreSQL on ARM : ecosystem, optimization & tuning

Bo Zhao

Amit Khandekar

Why ARM?

- Low cost of ownership
- All standard Linux operating systems available on ARM
- Many databases are available on ARM.
- Many powerful ARM chips available
    - Huawei: Kunpeng920
    - Amazon: AWS Graviton2
    - Ampere: Altra max 80-120 cores
    - Apple: M1 Laptop
    - Microsoft: News is that they are developing a ARM-based Chip..
- Growing adoption
- Widely available on Cloud

Speeding up PostgreSQL on ARM

- Apply Cost-performance model before comparing performance on ARM versus x86
  - https://mysqlonarm.github.io/CPM/
- Contention : Critical to scale out with very high number of clients
- Atomic operations should be optimized (__atomic_compare_exchange, __atomic_fetch_add)
  - These are Hardware and compiler dependent functions to manipulate memory atomically
- Look for platform-specific conditional compilation
- Compiler options and compiler versions
  - gcc 10.1 makes use of LSE instructions for atomic_compare_exchange, etc

Speeding up PostgreSQL on ARM

- NUMA awareness
- SIMD Vectorization
- CPU Cache line size
- Importance of Query Parallelism on ARM

When PostgreSQL bring ARM?

PG community provides the easiest way to use PostgreSQL on ARM64 platform.

2020.03

2020.04

PostgreSQL 9.5, 9.6, 10, 11, 12, 13 versions released Deb ARM64 packages for Debian and Ubuntu in Mar, 2020.

2020.08

aarch64 support is available on yum.postgresql.org

2020.09

PostgreSQL 9.5, 9.6, 10, 11, 12, 13 versions released RPM(YUM) ARM64 packages for RHEL, CentOS 7,8 and fedora in Aug, 2020.

ARM specific optimization

PG community provides the easiest way to use PostgreSQL on ARM64 platform.

2018.04	Optimize Arm64 crc32c implementation in Postgresql
2020.01	spin_delay() for ARM
2020.07	Inlining of couple of functions in pl_exec.c improves stored procedure performance
2020.09	[PATCH] audo-detect and use -moutline-atomics compilation flag for aarch64
	Auto-vectorization speeds up multiplication of large-precision numerics
2020.10	Improving spin-lock implementation on ARM.
2021.01	Speeding up GIST index creation for tsvectors

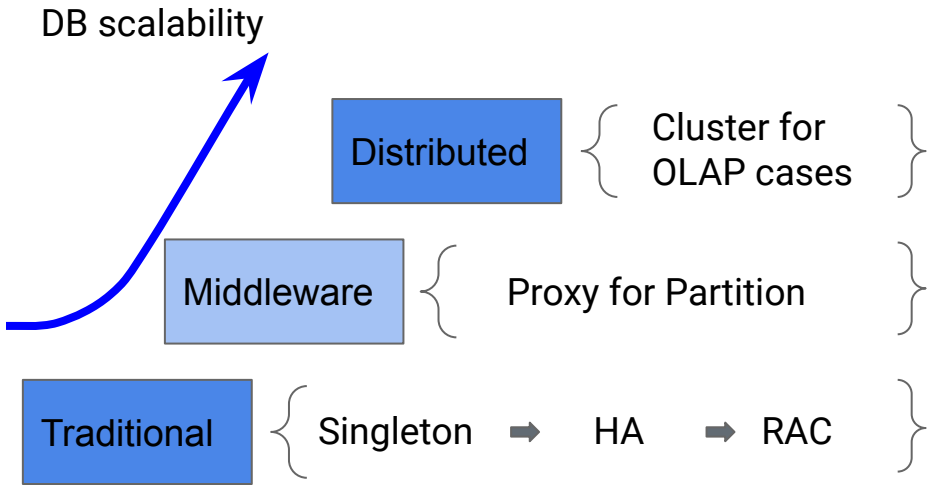
What we gain from ARM?

Adaption & optimization on ARM platform.

1. Inline of couple of functions in pl_exec.c improves stored procedure performance(7-14%)
2. spinlock optimization (10-40%)
3. SIMD optimization for multiplization of numeric types (2.7x)
4. GiST index build optimization (30-60%)
5. -moutline-atomics compilation flag for aarch64 (3%-10%)

What we gain from ARM?

More cores, more numa nodes and challenges.



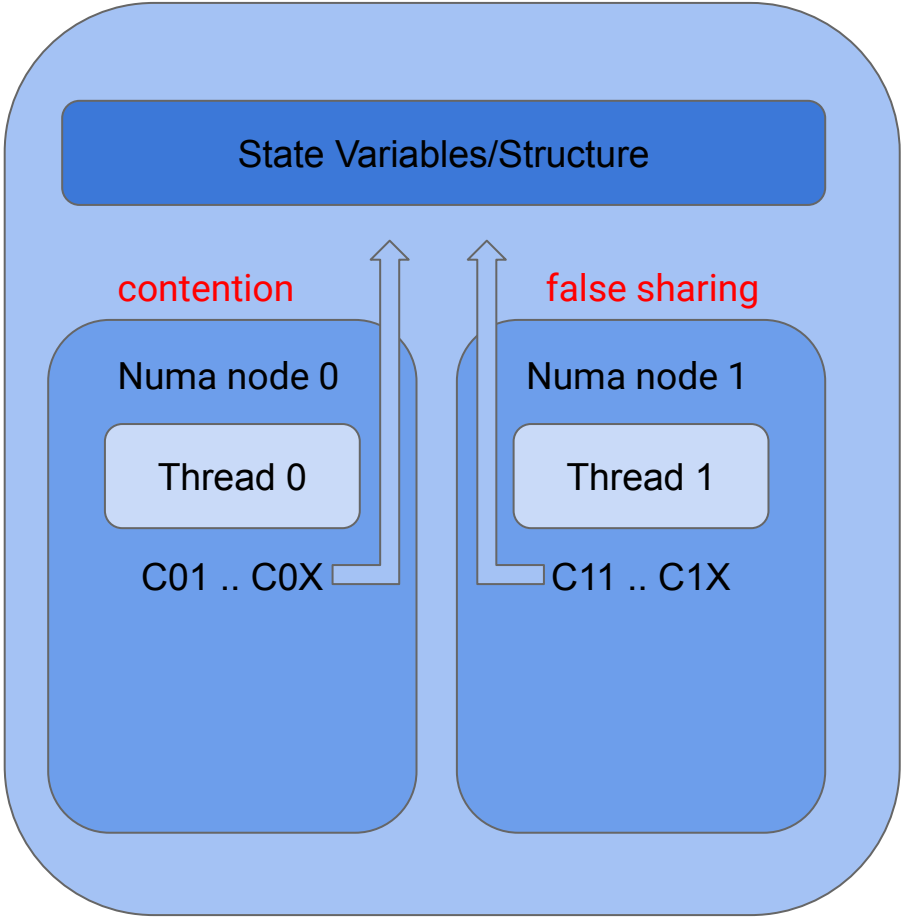
- ARM tend to have more Cores and in-turn more NUMA nodes.
- No hyper-threading support.

Challenges:

- Across Numa might bring bad/unstable performance.
- How to make a good use of these resources?

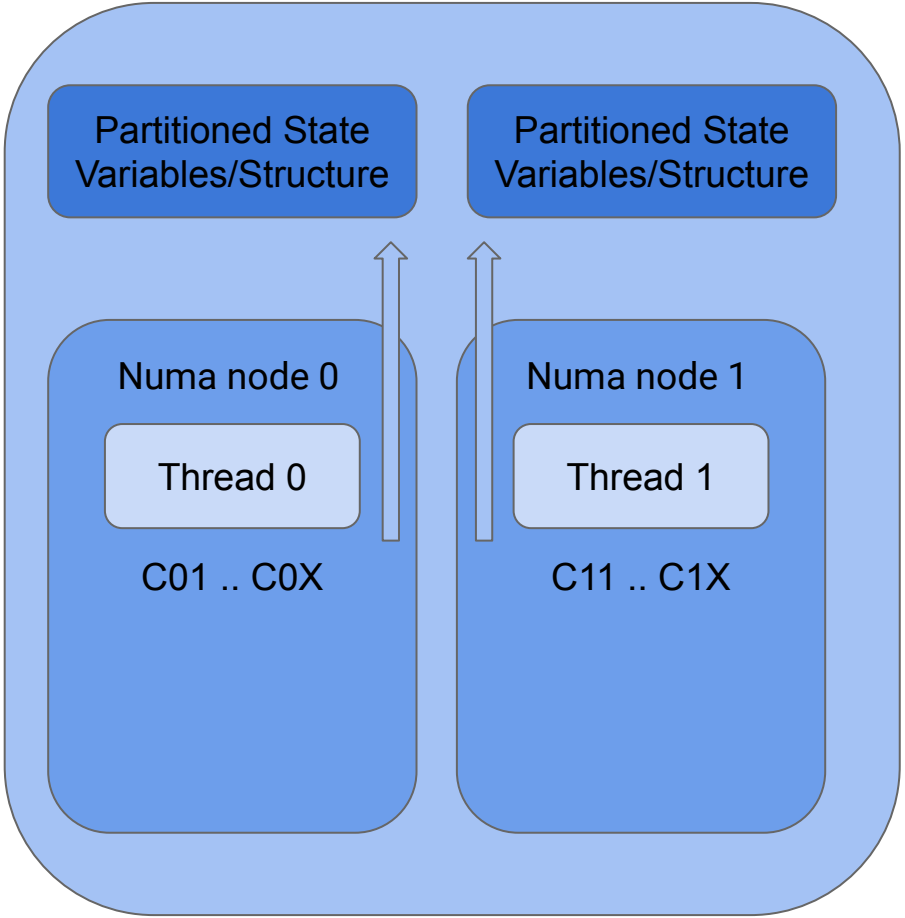
Numa Issue

Don’t access state variables/structure accross NUMA nodes.



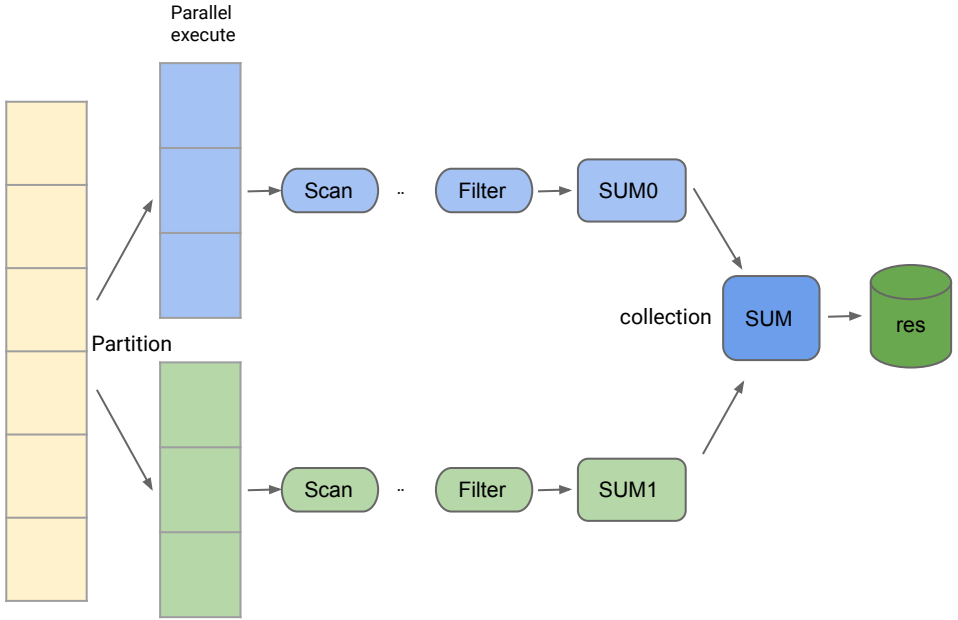
Numa Issue

Don’t access state variables/structure accross NUMA nodes.



Parallel Queries

- Large scale
- Resource utilization rate
- Bigdata scenarios



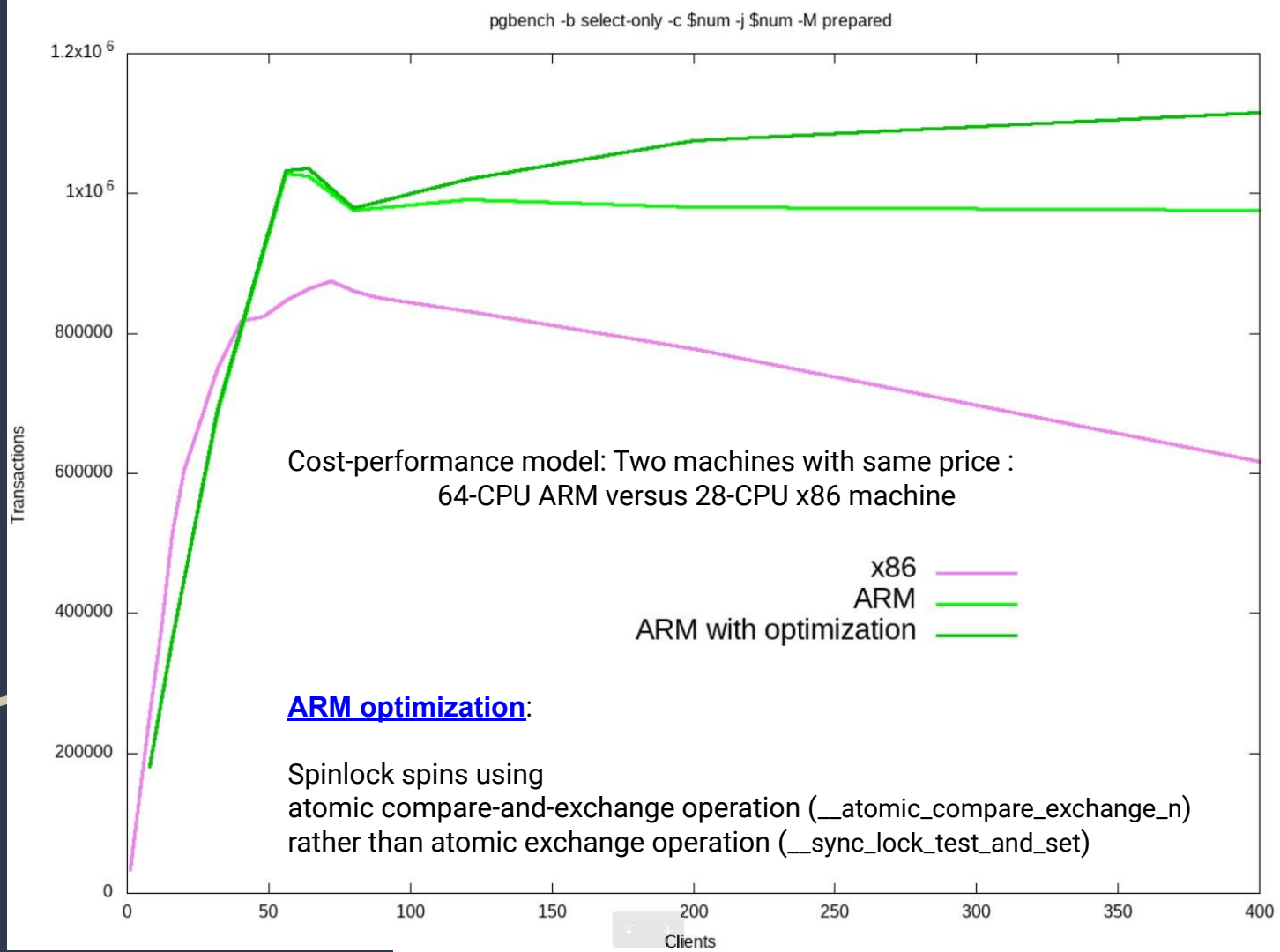
pgbench -b select-only

ARM64 :
Kunpeng 920 2.6 Hz
64 cores

x86_64 :
Intel(R) Xeon(R) Gold 6151
3.00 Gz
28 logical cores

Tables pre-warmed

pgbench scale=3000



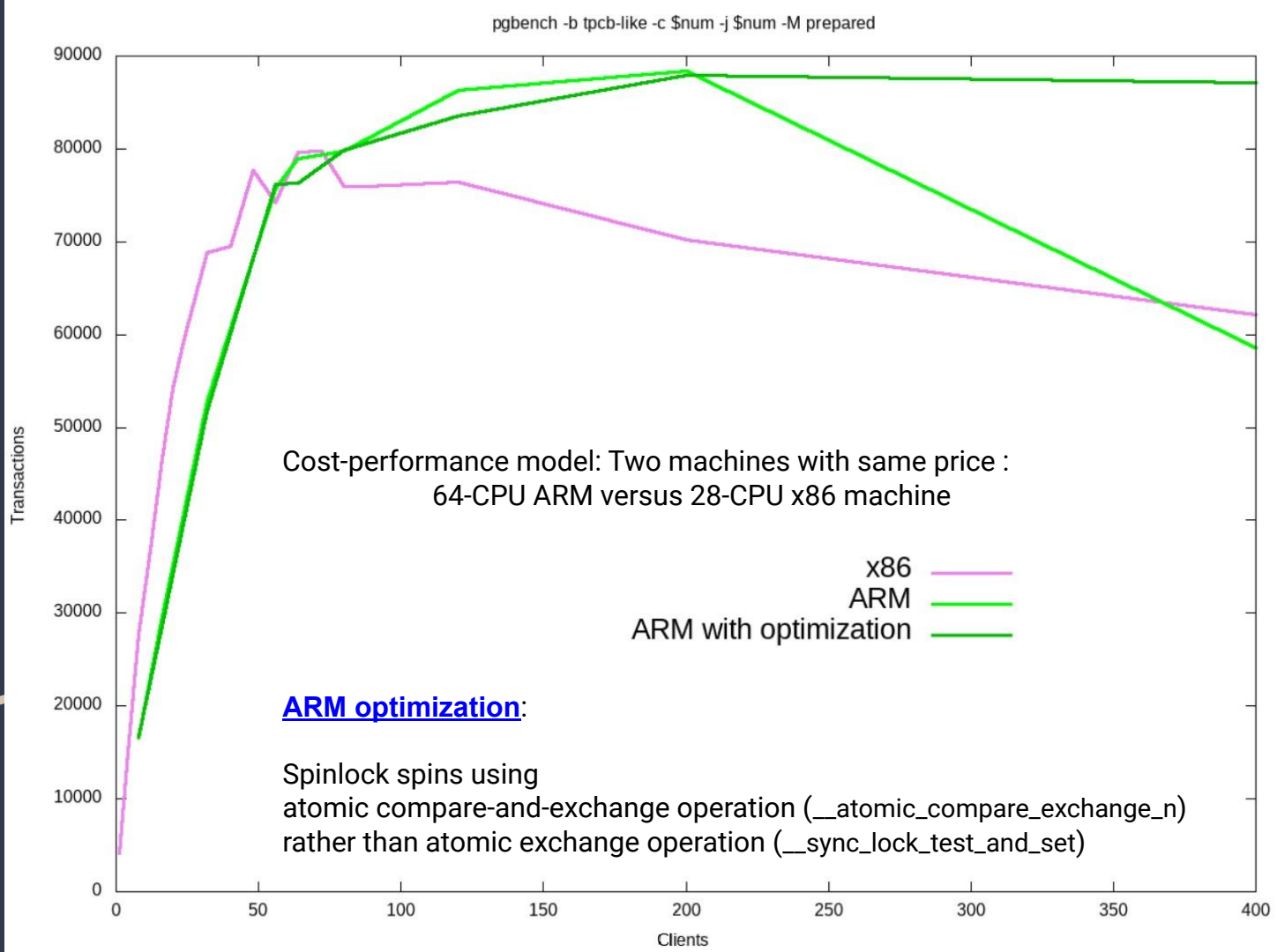
pgbench -b tpcb-like

ARM64 :
Kunpeng 920 2.6 Hz
64 cores

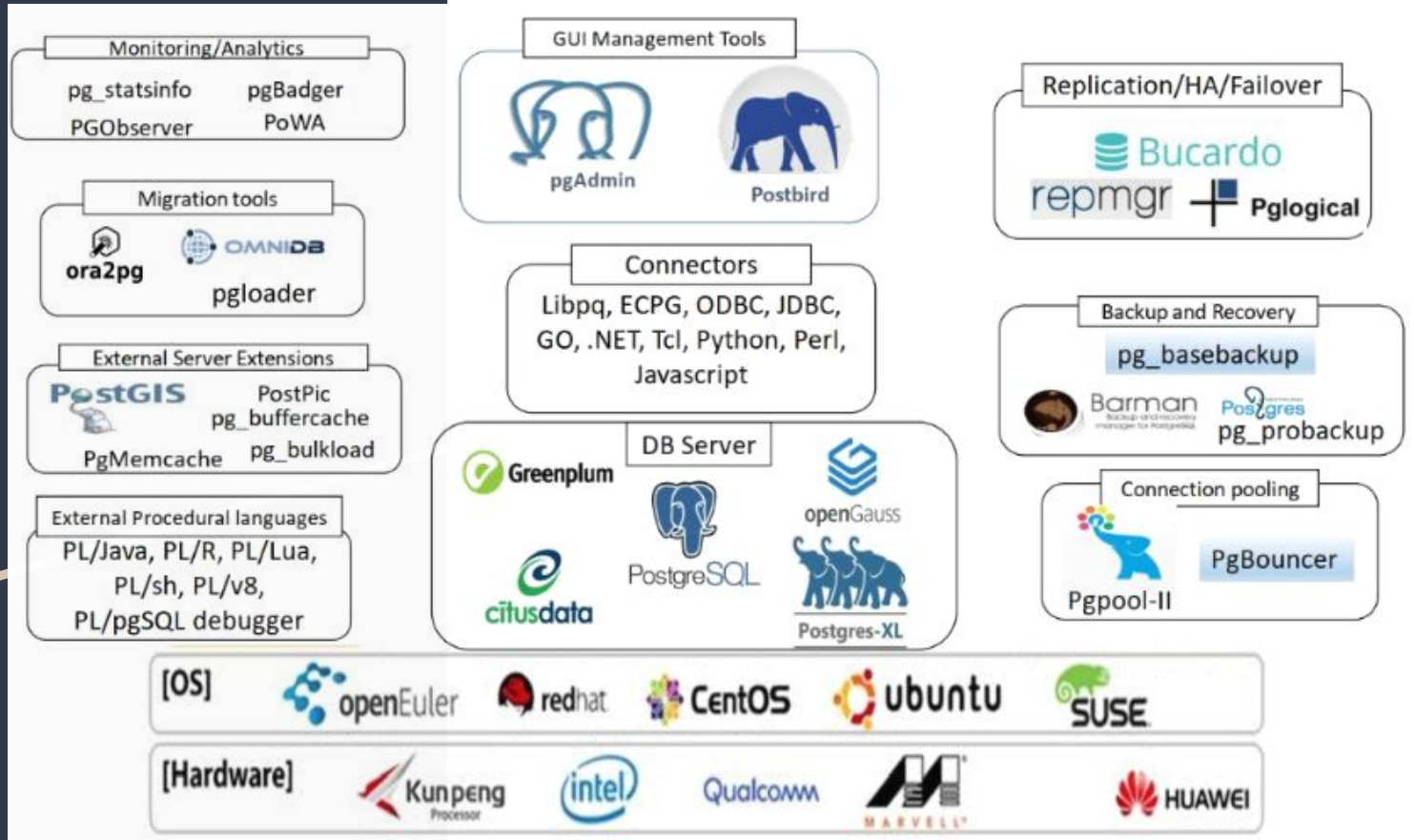
x86_64 :
Intel(R) Xeon(R) Gold 6151
3.00 Gz
28 logical cores

Tables pre-warmed

pgbench scale=3000



PostgreSQL ARM ecosystem



END