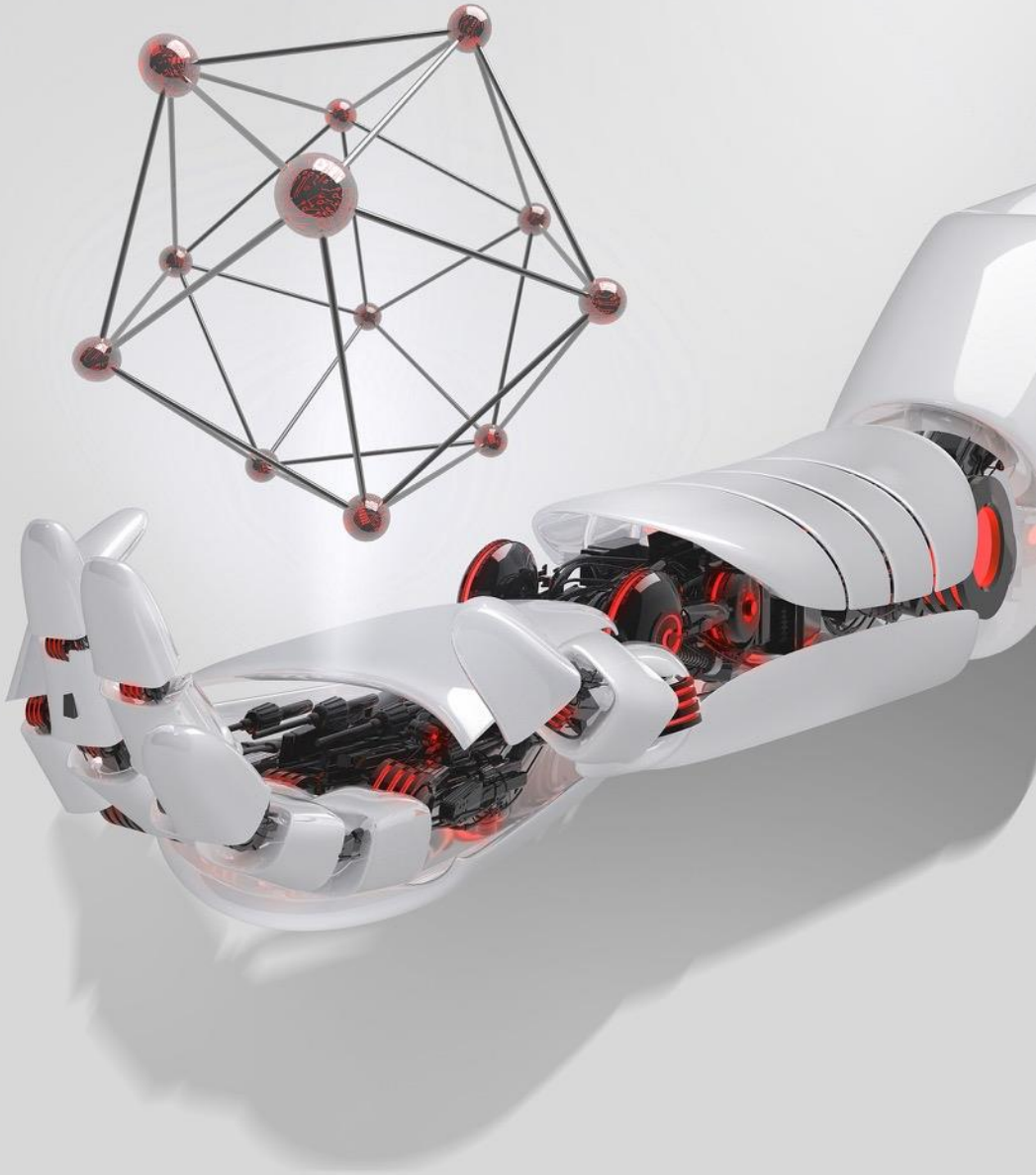
Boosting Application Performance on Arm Data Centers

- Zhenyu Zheng, Huawei Technologies



Linaro Connect VIRTUAL • SPRING 2021

HUAWEI

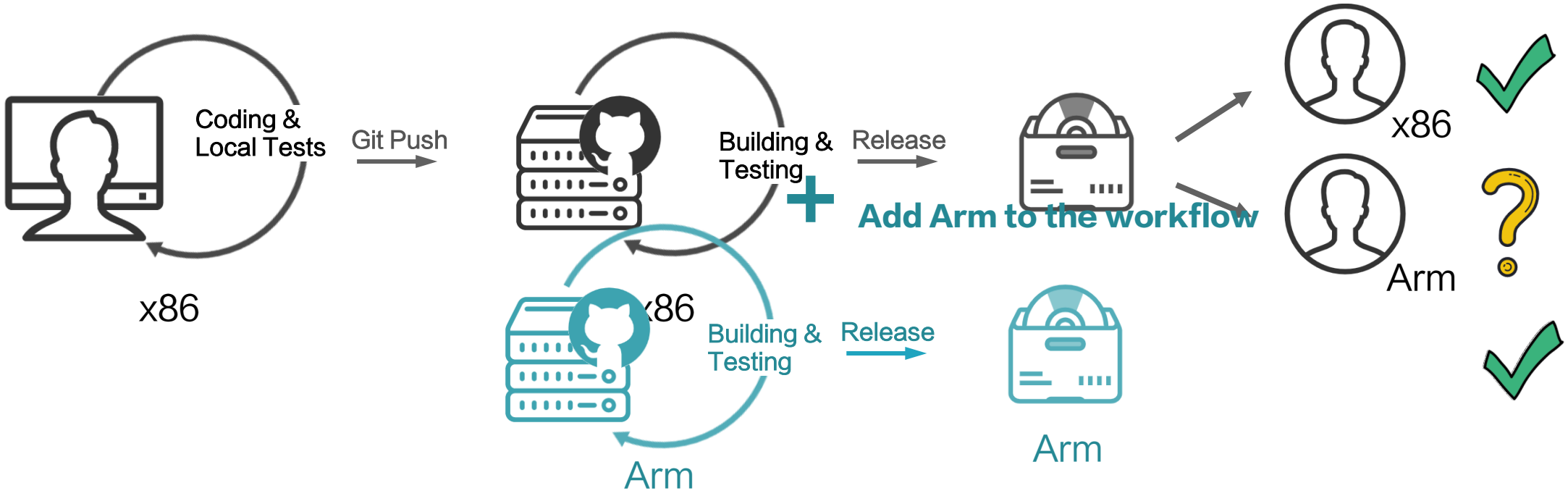
Contents:

- Arm CI recaps
- General Tunning for Arm Data Centers
- Vendor Specific Improvements

- Arm CI recaps

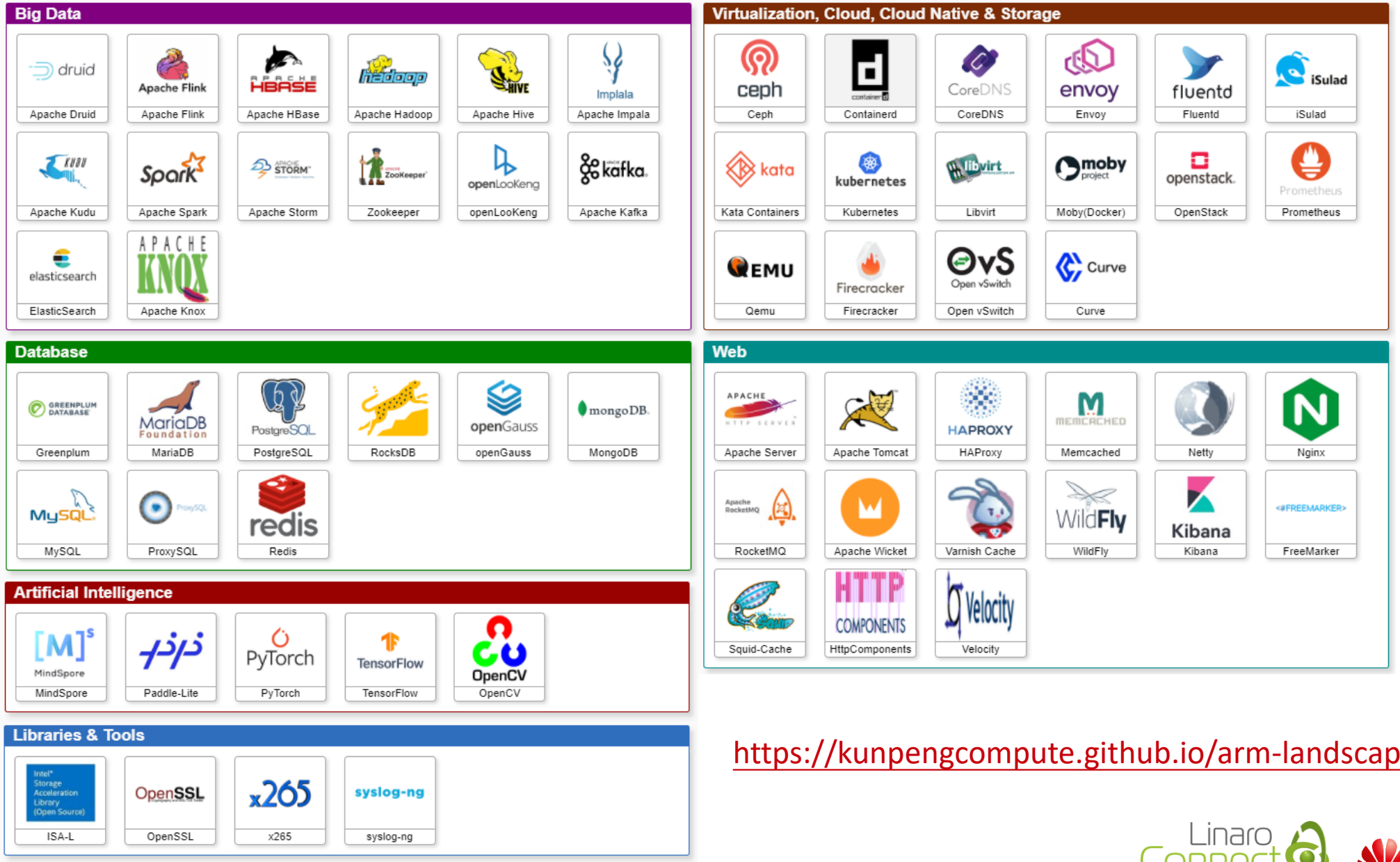
Dev workflow in Upstream

Dev workflow in Upstream



Linaro Connect VIRTUAL • SPRING 2021
HUAWEI

Arm CI landscape



https://kunpengcompute.github.io/arm-landscape/

Linaro Connect VIRTUAL • SPRING 2021

HUAWEI

Arm Available CI Resources

Donate

HUAWEI CLOUD

aws

Commercial Provider

Travis CI

Free Platform

openlab

DeveloperCloud
www.linaro.cloud

- General Tunning for Arm Data Centers

Spinlocks in DBs – TAS & CAS

*A **spinlock** is a `busy waiting` lock which causes a thread trying to acquire it to simply wait in a loop ("spin") while repeatedly checking if the lock is available.*

---- https://en.wikipedia.org/wiki/Spinlock

TAS(Test-and-Set): Instruction used to write 1 (set) to a memory location and return its old value as a single atomic operation

GCC: *__atomic_test_and_set()*

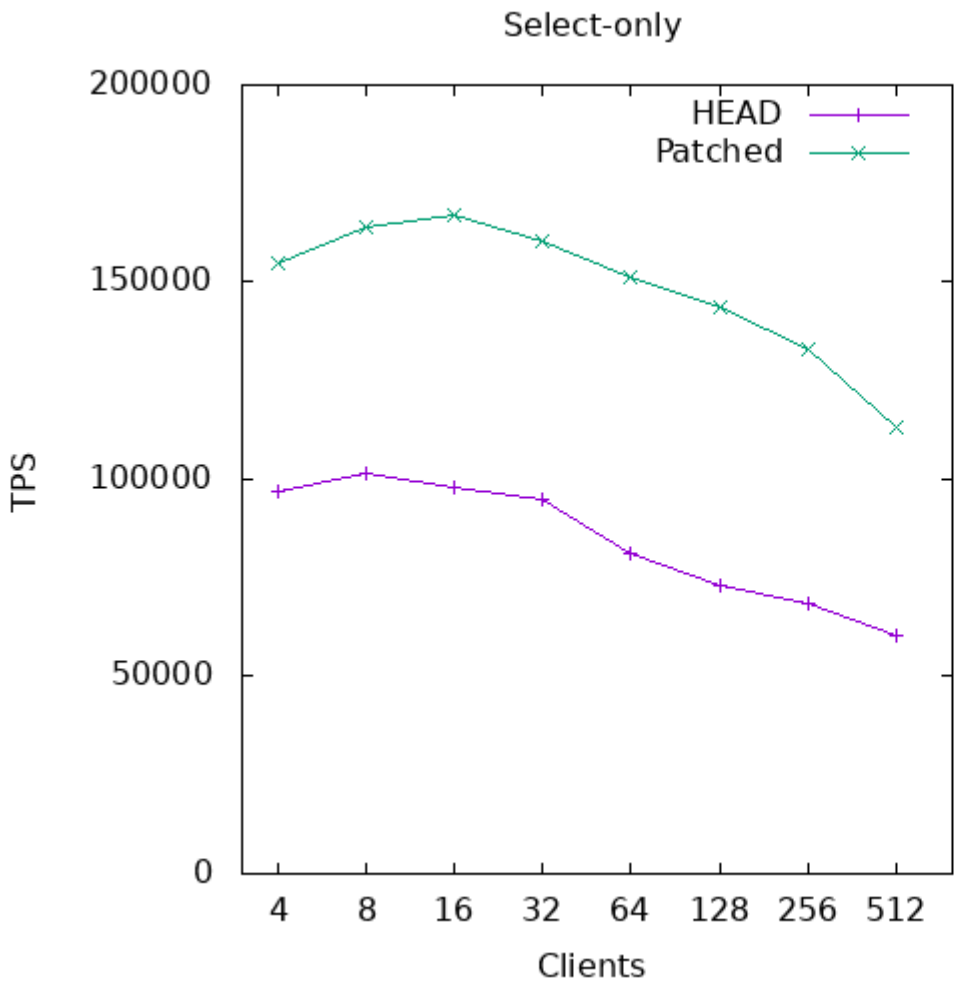
```
1   main:
2           sub     sp, sp, #16
3           mov     w0, 456
4           add     x2, sp, 12
5           str     w0, [sp, 12]
6           mov     w0, 1
7   .L2:
8           ldaxrb  w1, [x2]
9           stxrb   w3, w0, [x2]
10          cbnz    w3, .L2
11          mov     w0, 0
12          add     sp, sp, 16
13          ret
```

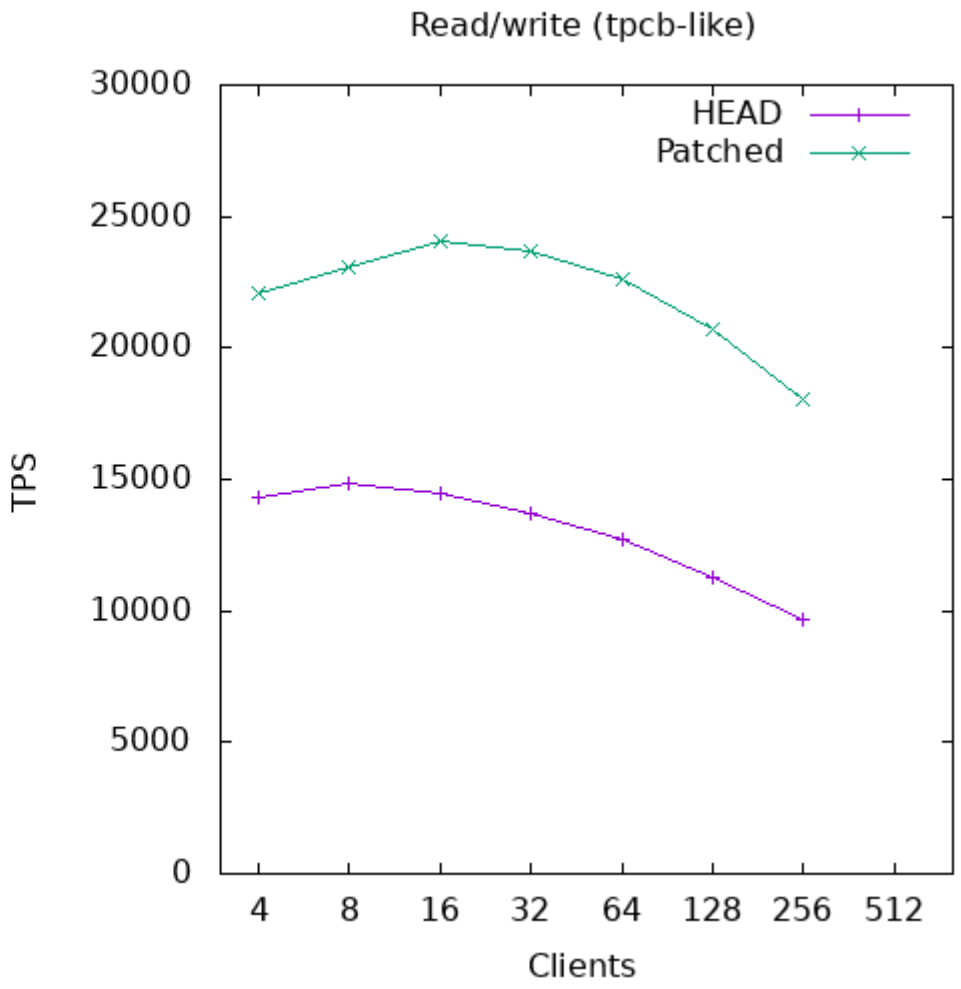
CAS(Compare-and-Swap): Atomic instruction compares the contents of a memory location with a given value and, only if they are the same, modifies the contents of that memory location to a new given value

GCC: *__atomic_compare_exchange_n ()*

```
1   main:
2           sub     sp, sp, #16
3           mov     w0, 456
4           add     x2, sp, 12
5           str     w0, [sp, 12]
6           mov     w0, 1
7   .L2:
8           ldaxr   w1, [x2]
9           cmp     w1, 123
10          bne     .L3
11          stxr    w3, w0, [x2]
12          cbnz    w3, .L2
13  .L3:
14          mov     w0, 0
15          add     sp, sp, 16
16          ret
```

Spinlocks in DBs – TAS & CAS





https://www.postgresql.org/message-id/flat/CAB10pyamDkTFWU_BVGeEVmkc8%3DEhgCjr6QBk02SCdJtKpHkdFw%40mail.gmail.com

Linaro Connect VIRTUAL • SPRING 2021
HUAWEI

include/my_cpu.h

Spinlocks in DBs – Delay

```
#ifdef _WIN32
  /*
    In the Win32 API, the x86 PAUSE instruction is executed by calling
    the YieldProcessor macro defined in WinNT.h. It is a CPU architecture-
    independent way by using YieldProcessor.
  */
  YieldProcessor();
#elif defined HAVE_PAUSE_INSTRUCTION
  /*
    According to the gcc info page, asm volatile means that the
    instruction has important side-effects and must not be removed.
    Also asm volatile may trigger a memory barrier (spilling all registers
    to memory).
  */
#ifdef __SUNPRO_CC
  asm ("pause" );
#else
  __asm__ __volatile__ ("pause");
#endif
#elif defined(_ARCH_PWR8)
  __ppc_get_timebase();
+ #elif defined __GNUC__ && (defined __arm__ || defined __aarch64__)
+   /* Mainly, prevent the compiler from optimizing away delay loops */
+   __asm__ __volatile__ (""":::"memory")
#else
  int32 var, oldval = 0;
  my_atomic_cas32_strong_explicit(&var, &oldval, 1, MY_MEMORY_ORDER_RELAXED,
                                  MY_MEMORY_ORDER_RELAXED);
#endif
}
```

12% performance improvement

https://github.com/MariaDB/server/commit/24f510bba4f0c3236aa5a8c96c209eb71317f4fe#

Using proper byte length for memcpy

```
1   diff --git a/strings/ctype.cc b/strings/ctype.cc
2   index 6fd5fb427b6..9dd11c728d4 100644
3   --- a/strings/ctype.cc
4   +++ b/strings/ctype.cc
5   @@ -942,7 +942,14 @@ size_t my_convert(char *to, size_t to_length, const CHARSET_INFO *to_cs,
6
7     length = length2 = MY_MIN(to_length, from_length);
8
9   -#if defined(__i386__) || defined(_WIN32) || defined(__x86_64__)
10  +#if defined(__x86_64__) || defined(__aarch64__)
11  + for (; length >= 8; length -= 8, from += 8, to += 8) {
12  +   if (uint8korr(from) & 0x8080808080808080) break;
13  +   int8store(to, uint8korr(from));
14  + }
15  +#endif /* defined(__x86_64__) || defined(__aarch64__) */
16  +
17  +#if defined(__i386__) || defined(_WIN32) || defined(__x86_64__) || defined(__aarch64__)
18    /*
19      Special loop for i386, it allows to refer to a
20      non-aligned memory block as UINT32, which makes
21
```

https://bugs.mysql.com/bug.php?id=98737

Linaro Connect VIRTUAL • SPRING 2021
HUAWEI

Hardware Accelerations – CRC32

Checksum:
Crc32: used for calculating table and binlog checksum
Crc32c: used in InnoDB for page-checksum

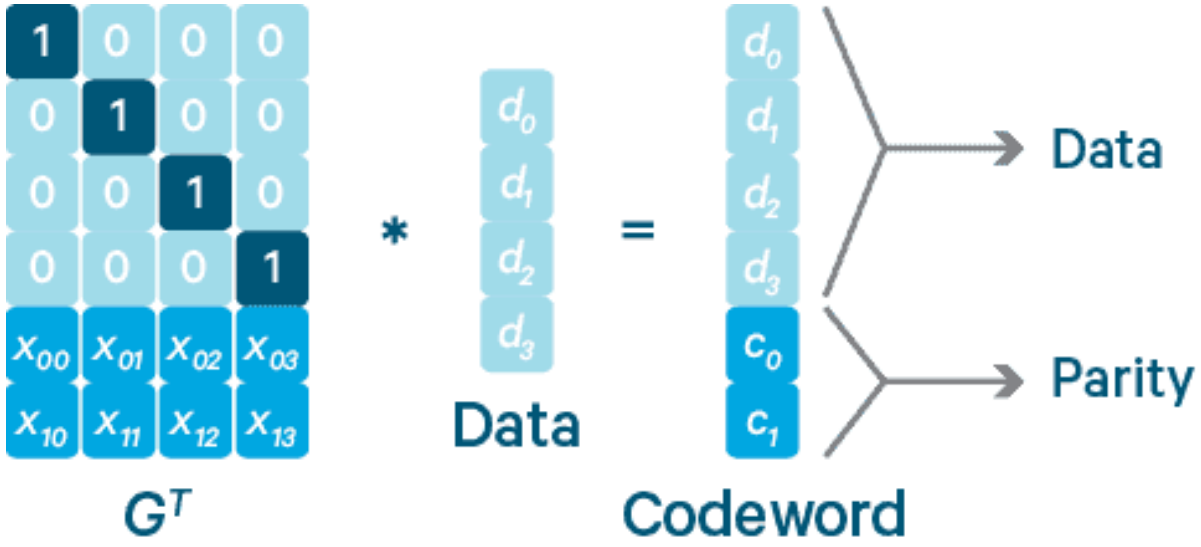
ACLE(Arm C Language Extensions) leverage the crc32 hardware acceleration and help reduces checksum time for the table. Our local testing revealed gain upto 50%(reduction in time):

Table-size(mil)	Table-checksum	Timing in sec	%reduction
----	w/o patch	w/ patch	----
0.1	0.15	0.06	60
1	1.5	0.69	54
5	7.55	3.55	53
10	14.94	6.09	54
50	75.25	34.5	54

https://bugs.mysql.com/bug.php?id=99118

Hardware Accelerations – NEON

Erasure Code: concept from coding theory, an erasure code is a **forward error correction (FEC)** code under the assumption of bit erasures (rather than bit errors), which transforms a message of k symbols into a longer message (code word) with n symbols such that the original message can be recovered from a subset of the n symbols.



Reed-Solomon encoding with four data units and two parity units

https://en.wikipedia.org/wiki/Erasure_code

Hardware Accelerations – NEON

HDFS by default replicates each block three times：

Pros:

- Replication provides a simple and robust form of redundancy to shield against most failure scenarios
- Eases scheduling compute tasks on locally stored data blocks by providing multiple replicas of each block to choose from

Cons:

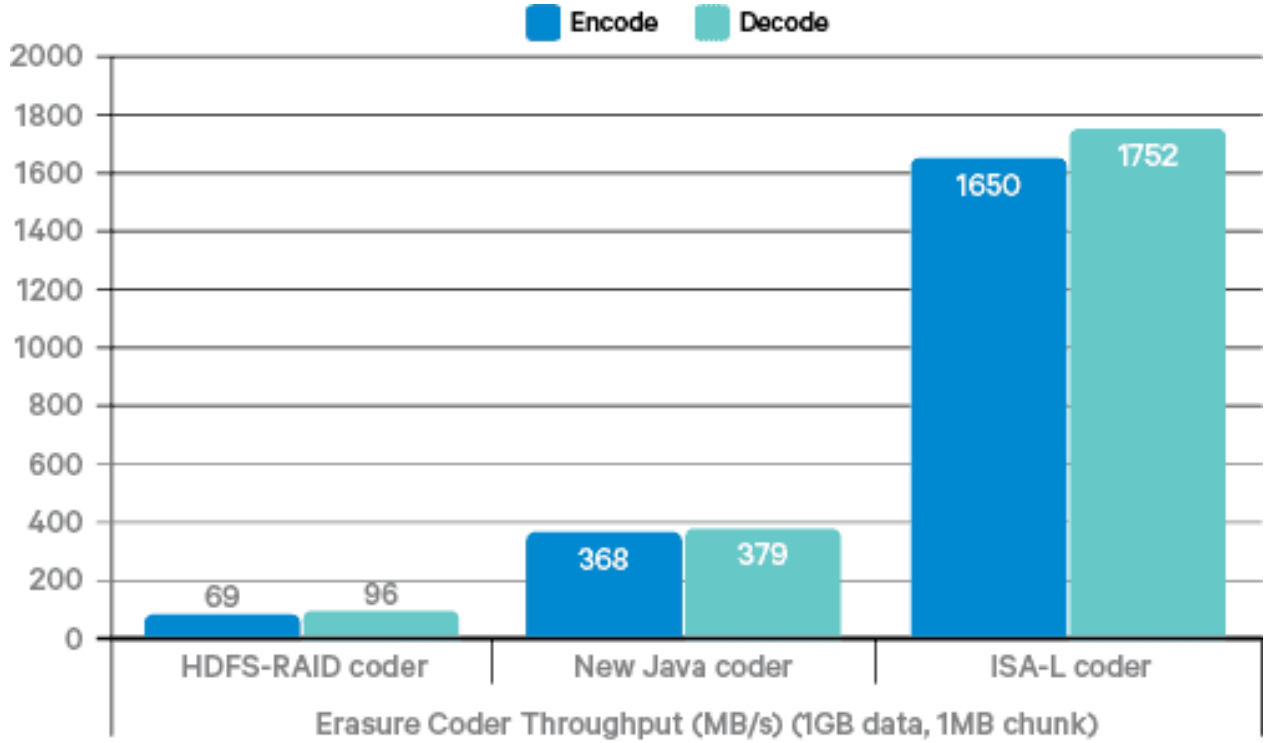
- Expensive, default 3x replication scheme incurs a 200% overhead in storage space and other resources
- For datasets with relatively low I/O activity, the additional block replicas are rarely accessed during normal operations, but still consume the same amount of storage space.

Solution:

Use erasure coding (EC) in place of replication, EC reduces the storage cost by ~50% compared with 3x replication under typical configuration

http://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-hdfs/HDFSErasureCoding.html

Hardware Accelerations – NEON



ISA-L: a collection of optimized low-level functions targeting storage applications, including EC, (de)compression etc.

Added NEON support for Arm platform in v2.28.0

https://github.com/intel/isa-l

Linaro Connect VIRTUAL • SPRING 2021
HUAWEI

Hardware Accelerations – Newer version of JDK

The problem：

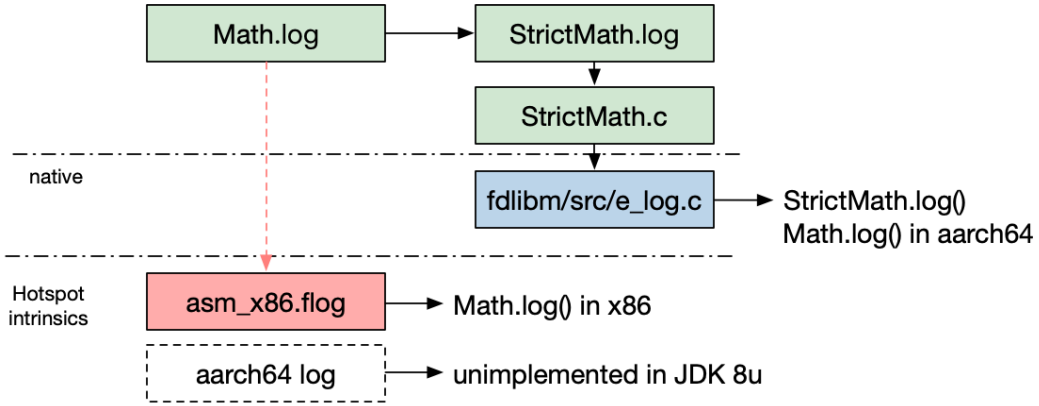
```
# on x86 – java 8
Math.log(3): 1.0986122886681098
StrictMath.log(3): 1.0986122886681096

# on aarch64 – java 8
Math.log(3): 1.0986122886681096
StrictMath.log(3): 1.0986122886681096
```

Not the same

Happens in Apache Hive, Spark, Flink

The Root Cause：



Math.log calls default implementation, it provided dynamic replacement for implementation on different architecture：

- x86 has assembly implementation，the precision is not guaranteed
- Arm64 has precise mathematical calculation，in JDK8 it uses common implementation，in JDK11 uses assembly with precision guaranteed.

Suggestion：

- Use newer version of JAVA to have better performance while maintain mathematical accuracy.

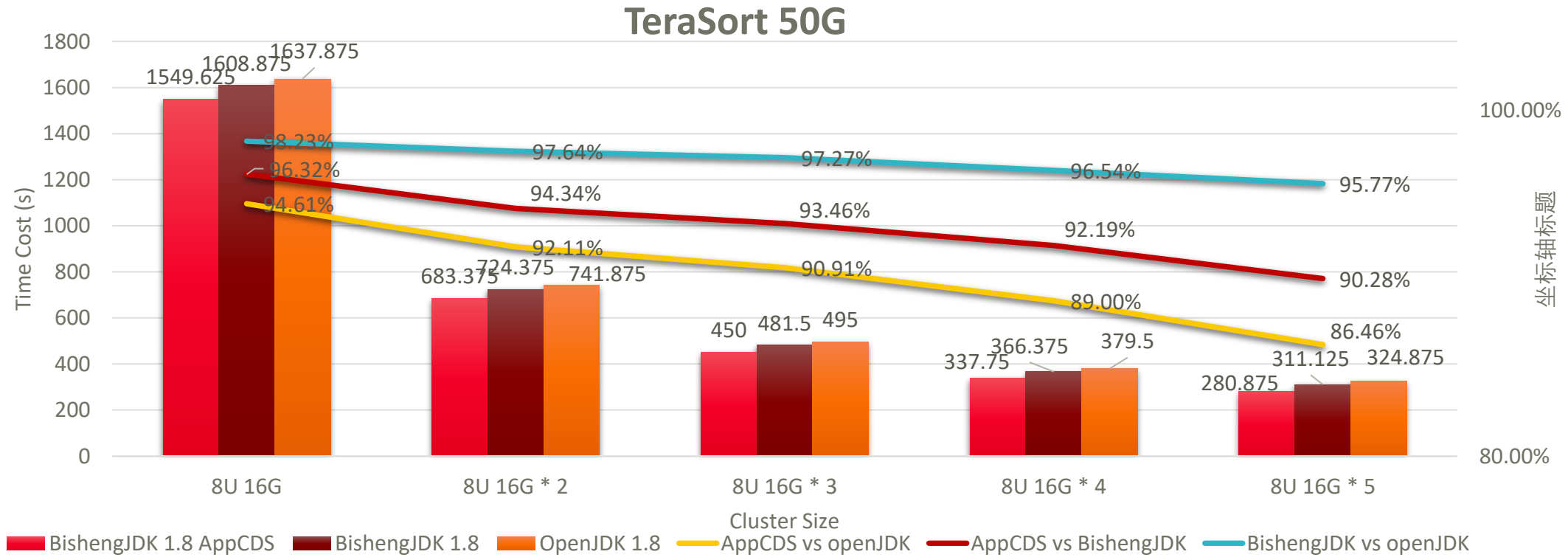
Linaro Connect VIRTUAL • SPRING 2021
HUAWEI

- Kunpeng Specific Optimizations

Kunpeng Specific Optimizations – BishengJDK

Bisheng JDK is a high-performance, production-ready downstream distribution of OpenJDK:

- Huawei used in over 500 products
- Accumulating a large number of usage scenarios and problems and demands from java developers
- Solving many problems encountered in the actual operation of the business
- Optimized for the ARM architecture



Kunpeng Specific Optimizations – KAE

Kunpeng Accelerator Engine(KAE) is a group of software libraries which provides performance competitiveness of common software on the Kunpeng platform. Including hardware-enabled accelerations for security, authentication, and compression.

So far, the algorithms supported by Kunpeng Accelerator Engine are:

Asymmetric encryption algorithm: RSA Support Key Sizes 1024/2048/3072/4096
Digest algorithm: SM3/MD5
Block cipher algorithm: SM4 Support CTR/XTS/CBC/ECB/OFB
Block cipher algorithm: AES Support CTR/XTS/CBC/ECB
Key exchange algorithm: DH Support 768bit/1024bit/1536bit/2048bit/3072bit/4096bit

https://github.com/kunpengcompute/KAE
https://github.com/kunpengcompute/KAEdriver

Kunpeng Specific Optimizations – KAE

```
[user1@kunpeng920 lzbench]$ ./lzbench -t16,16 -ezlib silesia.tar
lzbench 1.8 (64-bit Linux)  Assembled by P.Skibinski
Compressor name         Compress. Decompress. Compr. size  Ratio Filename
memcpy                7274 MB/s  7239 MB/s  211947520 100.00 silesia.tar
zlib 1.2.11 -1          55 MB/s  219 MB/s   77259029  36.45 silesia.tar
zlib 1.2.11 -2          51 MB/s  223 MB/s   75002277  35.39 silesia.tar
zlib 1.2.11 -3          42 MB/s  229 MB/s   72967040  34.43 silesia.tar
zlib 1.2.11 -4          38 MB/s  228 MB/s   71002817  33.50 silesia.tar
zlib 1.2.11 -5          28 MB/s  230 MB/s   69161685  32.63 silesia.tar
zlib 1.2.11 -6          20 MB/s  232 MB/s   68228431  32.19 silesia.tar
zlib 1.2.11 -7          16 MB/s  233 MB/s   67939647  32.05 silesia.tar
zlib 1.2.11 -8          10 MB/s  234 MB/s   67693427  31.94 silesia.tar
zlib 1.2.11 -9        8.27 MB/s  234 MB/s   67644548  31.92 silesia.tar
done... (cIters=1 dIters=1 cTime=16.0 dTime=16.0 chunkSize=1706MB cSpeed=0MB)
[user1@kunpeng920 lzbench]$ LD_PRELOAD=/usr/local/zlib/lib/libz.so.1.2.11 ./lzbench -t16,16 -ezlib silesia
lzbench 1.8 (64-bit Linux)  Assembled by P.Skibinski
Compressor name         Compress. Decompress. Compr. size  Ratio Filename
memcpy                7315 MB/s  7279 MB/s  211947520 100.00 silesia.tar
zlib 1.2.11 -1        2380 MB/s  1531 MB/s   97081116  45.80 silesia.tar
zlib 1.2.11 -2        2094 MB/s  1530 MB/s   97081315  45.80 silesia.tar
zlib 1.2.11 -3        2093 MB/s  1530 MB/s   97081388  45.80 silesia.tar
zlib 1.2.11 -4        2093 MB/s  1531 MB/s   97081235  45.80 silesia.tar
zlib 1.2.11 -5        2093 MB/s  1530 MB/s   97081838  45.80 silesia.tar
zlib 1.2.11 -6        1743 MB/s  1531 MB/s   97081749  45.80 silesia.tar
zlib 1.2.11 -7        2092 MB/s  1531 MB/s   97081527  45.80 silesia.tar
zlib 1.2.11 -8        2093 MB/s  1530 MB/s   97081553  45.80 silesia.tar
zlib 1.2.11 -9        2093 MB/s  1530 MB/s   97081604  45.80 silesia.tar
done... (cIters=1 dIters=1 cTime=16.0 dTime=16.0 chunkSize=1706MB cSpeed=0MB)
```

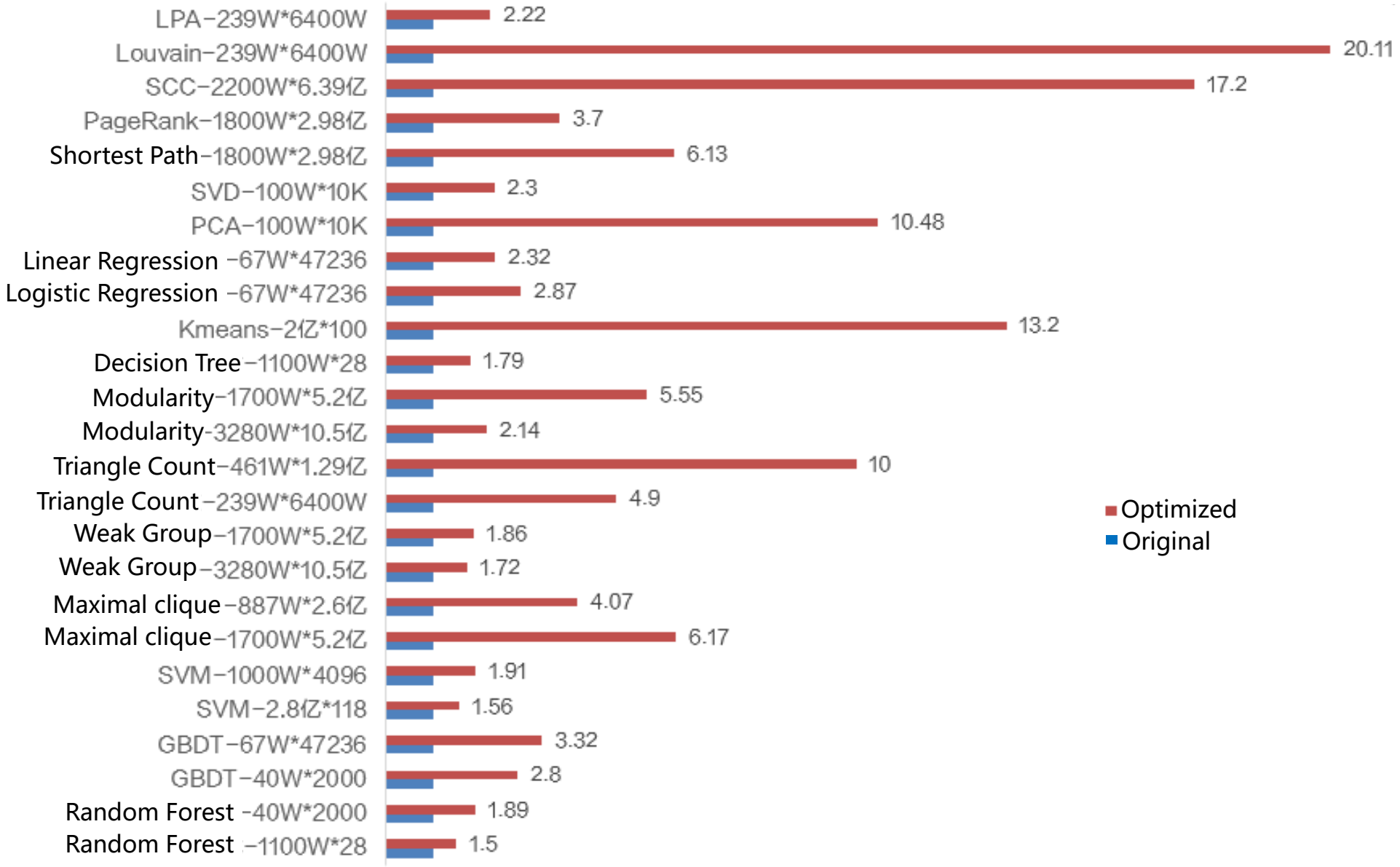
Kunpeng Specific Optimizations – Spark ML optimization

Kunpeng Spark MACHINE LEARNING Algorithm library is an accelerated library that provides a rich set of higher-level tools for machine learning algorithms.

Based on the original APIs from Apache Spark 2.3.2 and breeze 0.13.1. The accelerated library for performance optimization greatly improves the computational performance of big data algorithm scenarios.

Support Vector Machine (SVM) Algorithm
Random Forest Classifier (RFC) algorithm
Gradient Boosting Decision Tree (GBDT) algorithm
Decision Tree (DT) algorithm
Singular Value Decomposition (SVD) algorithm
K-means Clustering algorithm
Linear Regression algorithm,
Logistic Regression algorithm
Principal Component Analysis (PCA) algorithm

Kunpeng Specific Optimizations – Spark ML optimization



Thank You

Linaro Connect VIRTUAL • SPRING 2021
HUAWEI