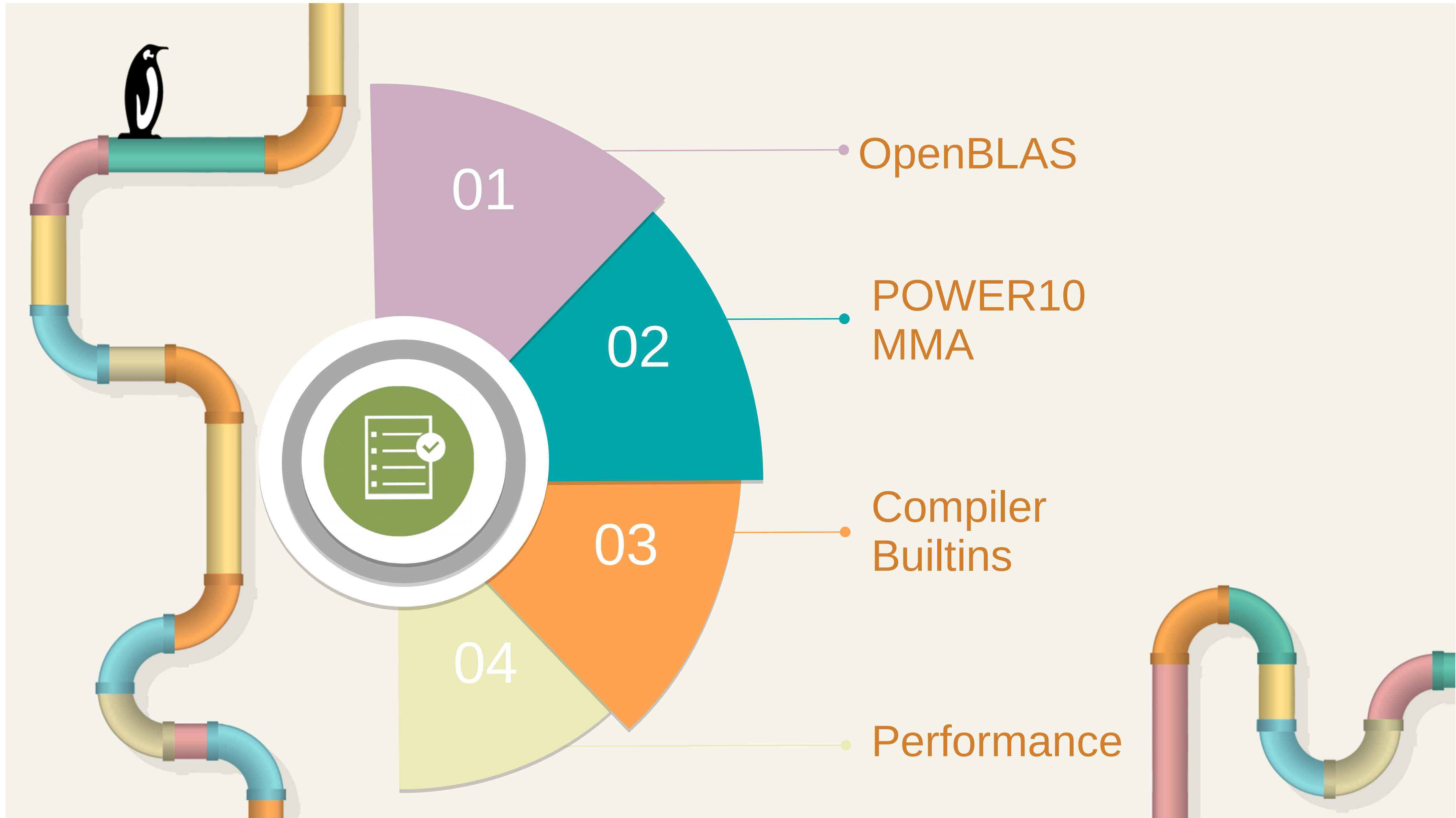
LINUX
PLUMBERS
CONFERENCE
August 24-28, 2020

Accelerating ML workloads using new GCC builtins


01
02
03
04
OpenBLAS
POWER10
MMA
Compiler
Builtins
Performance

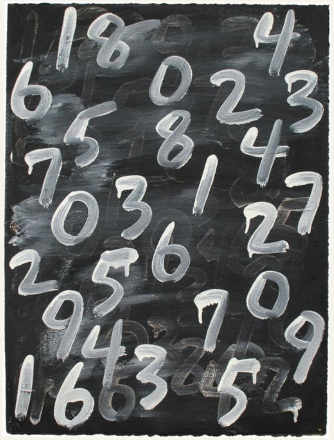
BLAS

The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations.

Level-1 BLAS perform scalar, vector and vector-vector operations

Level-2 BLAS perform matrix-vector operations

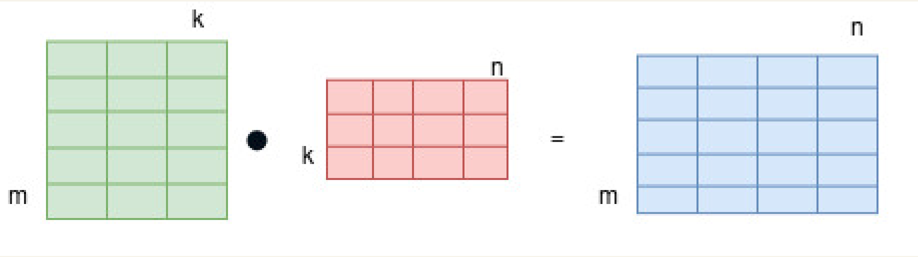
Level-3 BLAS perform matrix-matrix operations



OpenBLAS

An optimized BLAS library

- Optimized implementations of linear algebra kernels for several processor architectures.
- Default underlying library for many ML/DL frameworks.
- The source code distribution provides benchmarks for each BLAS kernel.
- Added **POWER10** support recently in OpenBLAS.
- Optimized GEMM [GEneral Matrix - Matrix] kernels.



POWER10 MMA

Most operations in training/inferencing in a neural network require some form of matrix multiplication.

Matrix Multiply Assist feature

- Eight 512 bit accumulators. Each accumulator contains four 128-bit rows

4X4 array of fp32-bit elements

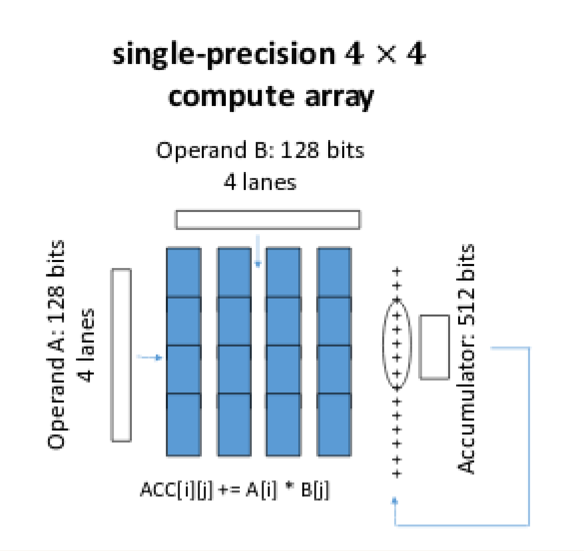
$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

4x2 array of fp64-bit

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix}$$

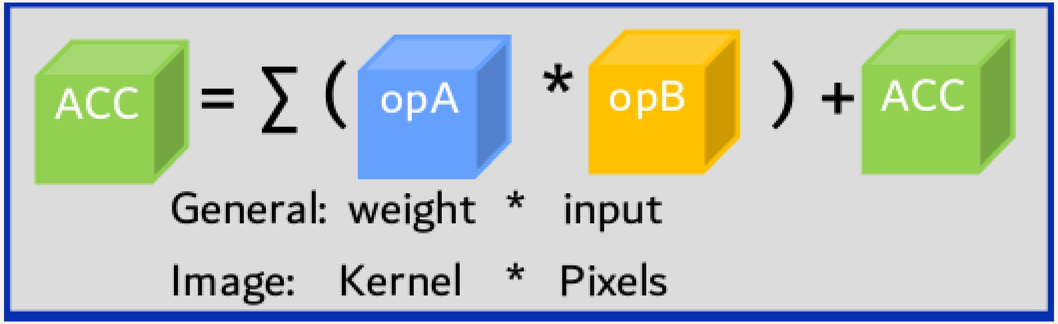
- A set of instructions to transfer data between vector-scalar registers and accumulators.
- A set of outer product instructions that perform an outer-product operation.

Outer-product (xv<type>ger<rank-k>) instructions



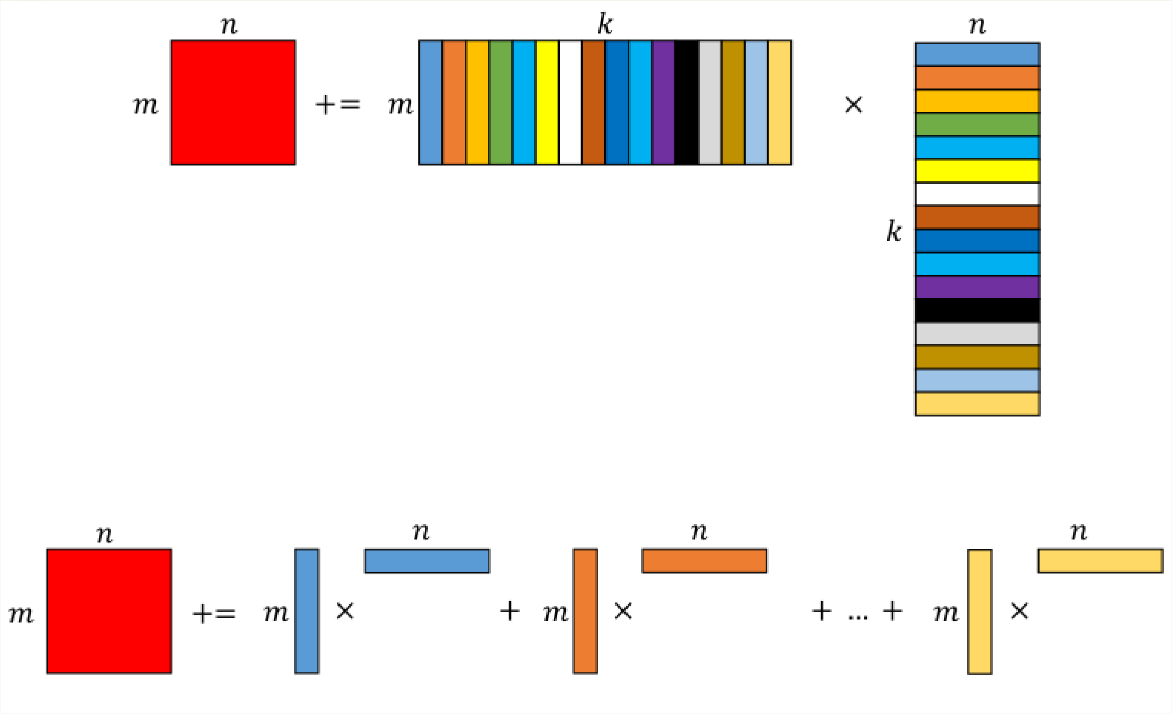
Accumulators are updated by rank-k update instructions:

- Input: 1 accumulator (A) + 2 VSRs (X, Y)
- Output: 1 accumulator (same as input to reduce instruction encoding space).



The **micro-kernel (innermost loop) of GEMM**

$C_{m\times n} \mathrel{+}= A_{m\times k} \times B_{k\times n}$



- Load a small, “square” panel of C and keep it in registers
- Load one small column of A and one small row of B
- Outer-product and accumulate
- Repeat!

Matrix-Multiply Assist Built-ins

- ISA 3.1 of the PowerPC added new Matrix-Multiply Assist (MMA) instructions
- GCC provides support for these instructions through the _builtin_mma_* built-in functions
- Some of the builtins used now in OpenBLAS are:

```
void __builtin_mma_xvbf16ger2 (__vector_quad *, vec_t, vec_t);
void __builtin_mma_xvf32ger (__vector_quad *, vec_t, vec_t);
void __builtin_mma_xvbf16ger2pp (__vector_quad *, vec_t, vec_t);
void __builtin_mma_xvf32gerpp (__vector_quad *, vec_t, vec_t);
void __builtin_mma_xvf64ger (__vector_quad *, __vector_pair, vec_t);
void __builtin_mma_xvf64gerpp (__vector_quad *, __vector_pair, vec_t);
void __builtin_mma_xxmtacc (__vector_quad *);
void __builtin_mma_xxmfacc (__vector_quad *);
void __builtin_mma_xxsetaccz (__vector_quad *);
void __builtin_mma_disassemble_acc (void *, __vector_quad *);
void __builtin_mma_disassemble_pair (void *, __vector_pair *);
```

Using new MMA builtins

- Hand written assembly version used in previous versions for GEMM optimization.
- Started POWER10 optimization with assembly and later converted to C code using built ins.
- Lines of code reduced from 6K to 1K for inner gemm kernels.
- Performance is closer to assembly version.
- These builtins are also now used in Eigen.

* Up to 4x improvements noted in simulator depending on various factors compared to previous processor.

LINUX PLUMBERS CONFERENCE

August 24-28, 2020

Thank You!

References:

- https://gcc.gnu.org/onlinedocs/gcc/PowerPC-Matrix-Multiply-Assist-Built-in-Functions.html
- https://github.com/xianyi/OpenBLAS/tree/develop/kernel/power