TensorFlow on state-of-the-art HPC clusters:
a machine learning use case

Guillem Ramirez-Gargallo
guillem.ramirez@bsc.es

Marta Garcia-Gasulla
marta.garcia@bsc.es

Filippo Mantovani*
filippo.mantovani@bsc.es

HPML workshop @ CCGRID19

Cyprus – 2019, May 14th

MONT-BLANC
EU-H2020 GA–671697

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

The context

The **Mont-Blanc** project

- European research project promoting / pushing Arm technology in HPC
- Initially focused on prototyping
- Interested in co-design with complex HPC applications

The **Student Cluster Competition**

- A team from BSC/UPC participates with Arm technology since 2015
- Teams have to operate a cluster within a power budget of 3 kW
- Since 2017 one of the challenges is focusing on AI (based on TensorFlow)

Contributions

- Study the performance of TensorFlow within the Arm hardware-software ecosystem
- Compare the performance of HPC clusters when running ML workloads

Approaches to “efficient” machine learning

Specialized hardware

- Tensor Processing Unit
- Graphics Processing Unit

Optimized linear algebra libraries

- Targeting different platforms/architectures
  e.g., MKL-DNN for x86

Research questions

Data centers are powered by different architectures (e.g., x86, Power9, Arm)

- How current HPC architectures behave under ML workloads?
- Which is the impact of optimized arithmetic libraries when running ML workloads?
- How to increase the efficiency of homogeneous clusters with ML workloads?

The hardware platforms – MareNostrum4, Intel x86

- #25 in the Top500 (Nov 2018)
- 11.15 PetaFlops peak performance
- 384.75 TB of main memory
- 3456 nodes:
  - 2× Intel Xeon Platinum 8160 (24 cores at 2.1 GHz)
  - 6 DDR4 memory channels at 2667 MHz per socket
  - 216 nodes with 384 GB/node
  - 3240 nodes with 96 GB/node
- Interconnection networks:
  - 100Gb Intel Omni-Path Full-Fat Tree
  - 10Gb Ethernet

The hardware platforms – Power9, IBM Power

Same architecture of Summit
#1 of the Top500

- #411 in the Top500 (Nov 2018)
- 1.5 PetaFlops peak performance
- 52 compute nodes, each of them:
  - 2× IBM Power9 8335-GTH @ 2.4 GHz (3.0 GHz on turbo, 20 cores and 4 threads/core)
  - 512 GB of main memory distributed on 32 GB × 16 DIMMS @ 2666 MHz
  - 2× SSD 1.9 TB as local storage
  - 2× 3.2 TB NVME
  - 4× GPU NVIDIA V100 (Volta) with 16 GB HBM2.
- Interconnection network: Single Port Mellanox EDR

The hardware platforms – Dibona, Armv8

Same architecture of the Astra (Sandia)
#204 of the Top500

- 49 TeraFlops theoretical peak performance
- 48 nodes, each node includes:
  - 2× Armv8 Marvell ThunderX2 CPUs @ 2.0 GHz (2.5 GHz on turbo, 32 cores and 4 threads/core)
  - 8 memory channels per socket
  - 256 GB DDR4 memory
  - 128 GB SSD local storage
- Interconnection network: Single Port Mellanox EDR

Machine learning environment

- Training phase of TensorFlow
- Image recognition based on ImageNet
  - Synthetic image set to avoid I/O overhead
  - Real image set for scalability tests
- Two models
  - AlexNet
  - ResNet-50

Figures of interest

- *Performance* expressed as *img/s*
- *Batch size* and its effect on the training performance
- *Training effectiveness* checking the increasing trend of the accuracy

Methodology

We evaluate the *vanilla* version of TensorFlow and the one using vendor specific libraries

Cluster	TF version	Compiler	MPI	Back-End / Perf. Libs.	Front-End Flags
MareNostrum4	r1.11	GCC 7.2.0	OpenMPI 3.1.1	Vanilla/Eigen	-mtune=skylake-avx512 -march=skylake-avx512 -O3
MareNostrum4	r1.11	GCC 7.2.0	OpenMPI 3.1.1	MKL DNN v0.16	-mtune=skylake-avx512 -march=skylake-avx512 -O3
Dibona	r1.11	GCC 8.2.0	OpenMPI 2.0.2.14	Vanilla/Eigen	-march=native -mtune=thunderx2t99 -O3
Dibona	r1.11*	GCC 8.2.0	OpenMPI 2.0.2.14	ArmPL 19.0	-march=native -mtune=thunderx2t99 -O3
Power9	r1.11	GCC 8.2.0	OpenMPI 3.1.1	Vanilla/Eigen	-mtune=power9 -mcpu=power9 -O3

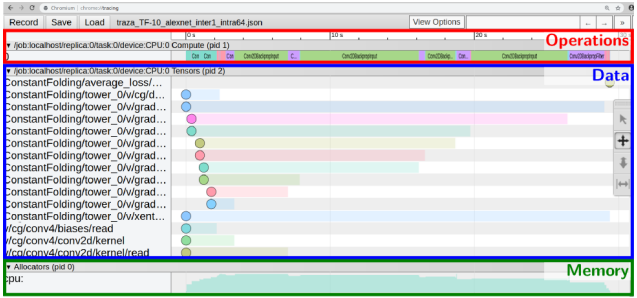
Single-node evaluation

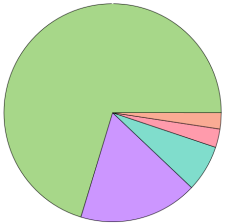
- Study of multi-threaded parallelism varying the batch size
  - Changing the number of threads used by each kernel
- Study of multi-process parallelism vs. multi-threaded paralellism
  - Leveraging Horovod

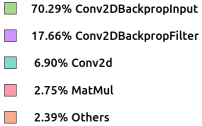
Multi-node evaluation

- Study of performance at scale

Profiling of operations – TensorFlow kernels





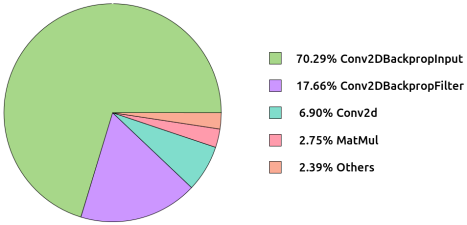


Parameters

- AlexNet model
- Syntetic image dataset based on ImageNet
- Batch size = 1024
- Intra-ops = 64
- Inter-ops = 1

Profiling of operations – Underlying libraries

Overhead	Shared object	Symbol
76.58%	_pywrap_tensorflow_internal.so	Eigen::internal::gebp_kernel<float, float, long,...
4.44%	_pywrap_tensorflow_internal.so	Eigen::internal::gemm_pack_rhs<float, long,...
4.06%	_pywrap_tensorflow_internal.so	Eigen::internal::gemm_pack_rhs<float, long,...
2.02%	_pywrap_tensorflow_internal.so	std::_Function_handler<void (long, long),...
1.50%	_pywrap_tensorflow_internal.so	Eigen::internal::gemm_pack_lhs<float, long,...
1.42%	_pywrap_tensorflow_internal.so	(anonymous namespace)::Col2im<float>
1.32%	libc-2.17.so	memcpy



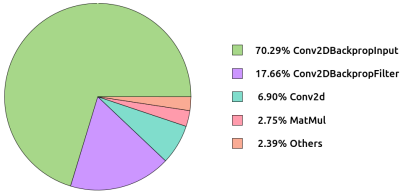
Parameters

- AlexNet model
- Syntetic image dataset based on ImageNet
- Batch size = 1024
- Intra-ops = 64
- Inter-ops = 1

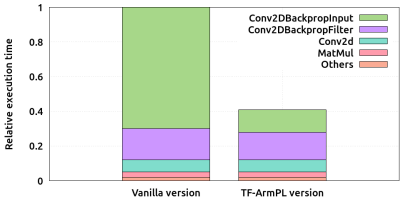
Improve performance plugging the Arm Performance Libraries

- Since 2016 Arm releases the *Arm Performance Libraries* a collection of libraries including BLAS, LAPACK, FFT and standard math routines.
- There was no version of TensorFlow leveraging the Arm Performance Libraries.
- We did it and share it with Arm. They are going to realease it here: `https://gitlab.com/arm-hpc/packages/wikis/packages/tensorflow`

```
#ifdef ARMPL
  auto out_data = out_backprop_data + output_offset * image_id;
  contract_to_cblas<T>( context, false, true, out_data, filter_data,
    col_buffer_data, output_image_size, filter_total_size, dims.out_depth, false);
#else
  C.device(context->eigen_cpu_device()) = A.contract(B, contract_dims);
#endif //ARMPL
```



⇒



Intranode evaluation

We evaluate the **performance of TensorFlow** within a single computational node

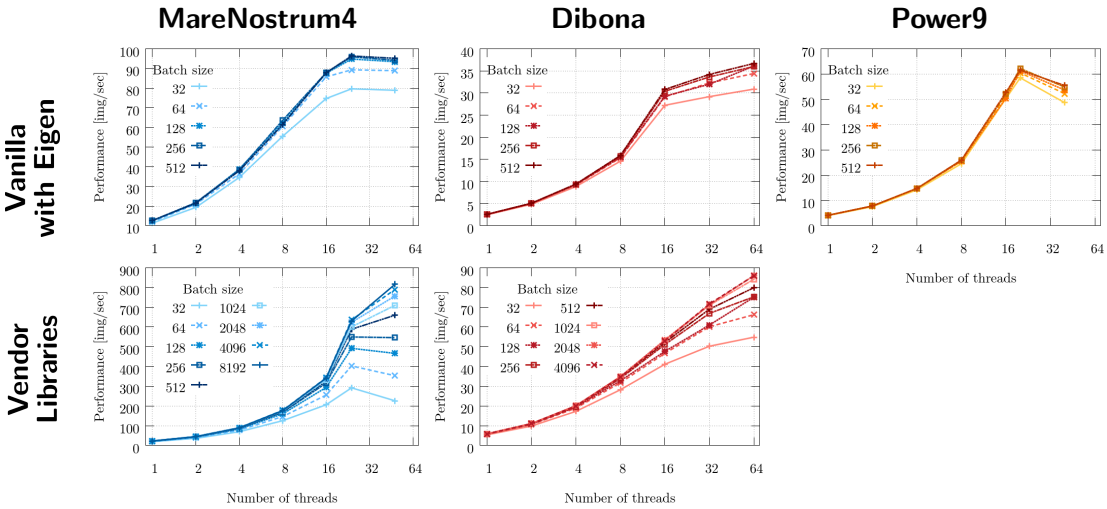
We divide the study into two parts:

1. We study the effect on the performance [img/s] when
   - **1.1** changing the **number of threads**
   - **1.2** varying the **batch size**
2. We evaluate the **best configurations of MPI processes and threads** when using all the computational resources of one node

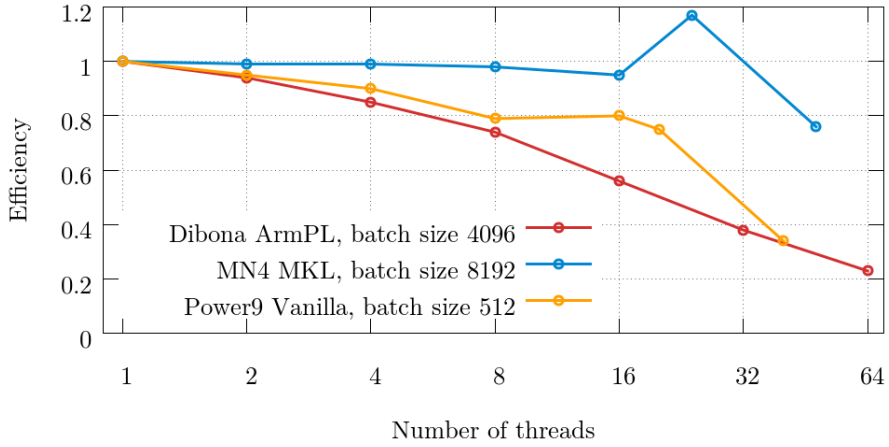
We perform both studies on **MareNostrum4**, **Power9** and **Dibona** leveraging as backend

- **Eigen**
- **Vendor-specific** linear algebra **libraries** (when available)
  MKL for x86 and Arm Performance Libraries for Arm

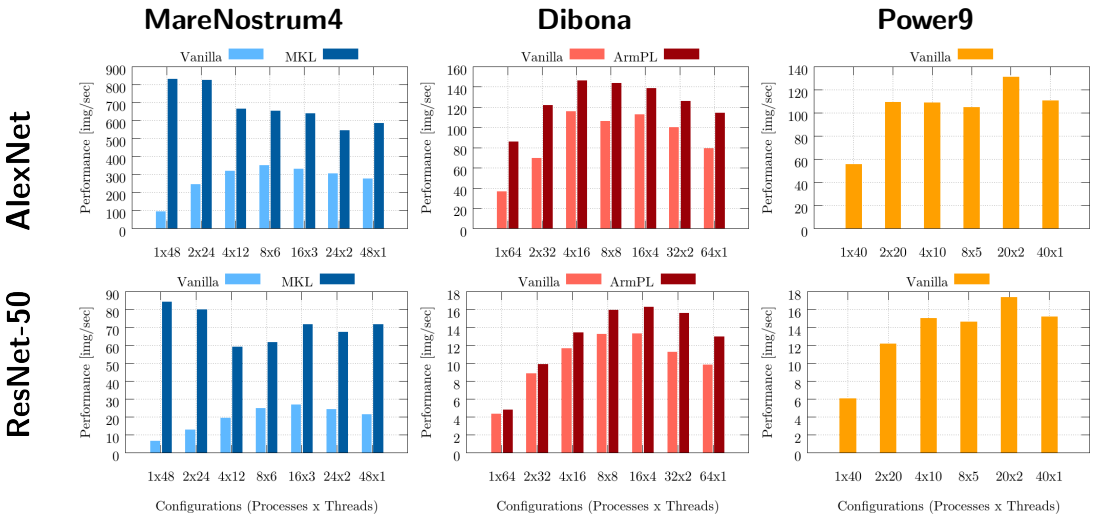
Thread scaling – AlexNet



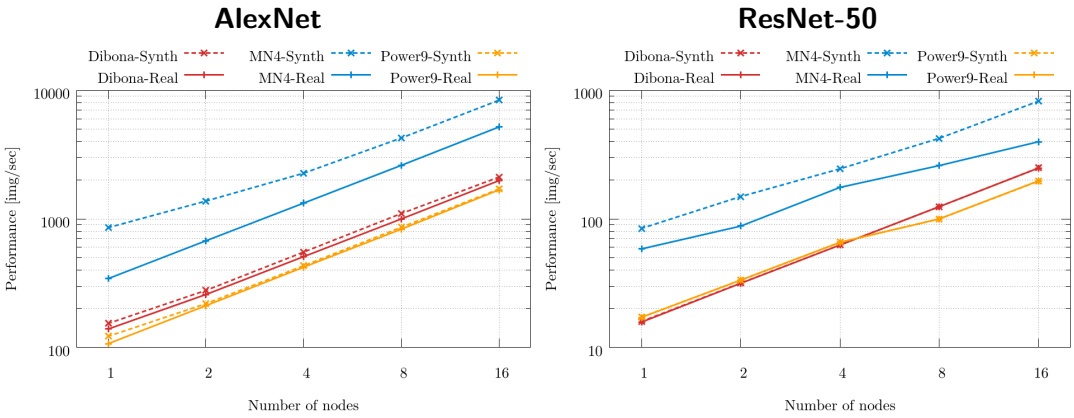
Thread scaling – AlexNet – efficiency



Hybrid Configurations Evaluation



Scalability



Conclusions

- Contribution to the Arm ecosystem developing a version of TensorFlow with Arm PL
- Classical HPC optimization techniques are beneficial also for ML
- Evaluation of TensorFlow with AlexNet and ResNet-50 on HPC clusters
  - using ArmPL on latest Marvell's Arm CPU (ThunderX2)
    1.5× to 2.3× speedup compared to Vanilla
  - using MKL on latest Intel x86 CPU (Skylake)
    1.7× to 6.9× speedup compared to Vanilla
  - using Eigen on latest IBM Power9 CPU

More details can be found in: `https://upcommons.upc.edu/handle/2117/131762`

Guillem Ramirez-Gargallo
guillem.ramirez@bsc.es

Marta Garcia-Gasulla
marta.garcia@bsc.es

Filippo Mantovani
filippo.mantovani@bsc.es