Toward a BLAS library truly portable across different accelerator types

Eduardo Rodriguez-Gutiez, Ana Moreton-Fernandez, Arturo Gonzalez-Escribano, Diego R. Llanos Universidad de Valladolid, Spain

{eduardo|arturo|ana|diego}@infor.uva.es

Universidad de Valladolid

Grupo Trasgo

Universidad de Valladolid

Introduction

- Linear algebra kernels are in the core of many scientific applications.
- BLAS: Basic Linear Algebra routines.
- Several specialized implementations: Intel MKL, MAGMA, cuBLAS, etc.
- Differences in interface and usage (context, memory transfers, etc.)
- Programs that can adapt to the platform at run-time?
  - Programmer should deal with differences and hardwire in the code the use of different libraries for different devices.
- Contributions:

(1) A unified, performance-oriented, and portable interface for BLAS

(2) Integrated in the Controller programming model

(3) Support different types of devices: GPUs, CPUs, XeonPhi

(4) Hide interface and usage differences: IntelMKL, cuBLAS, MAGMA, etc.

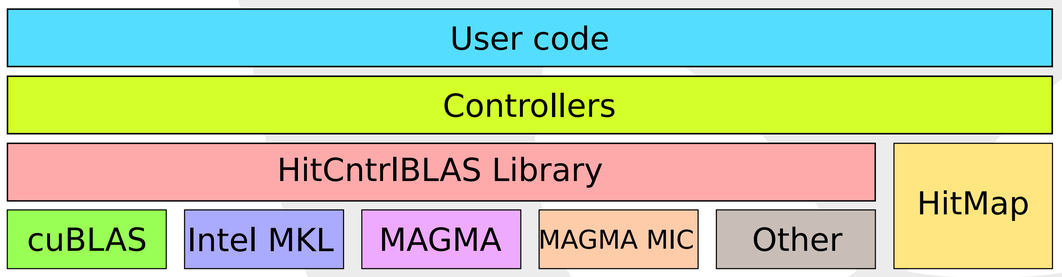
(5) Selection policy: Choose the most appropriate available library

(6) Flexibility, interoperability with programmer defined kernels

(7) Code and performance portability

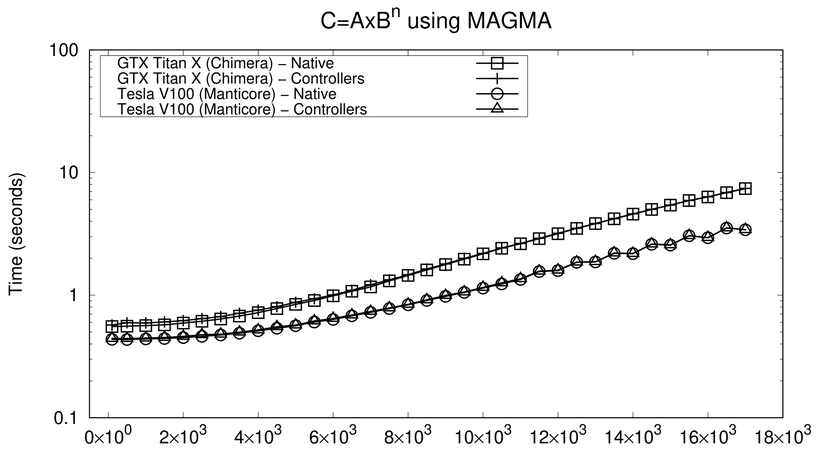
HitCtrlBLAS

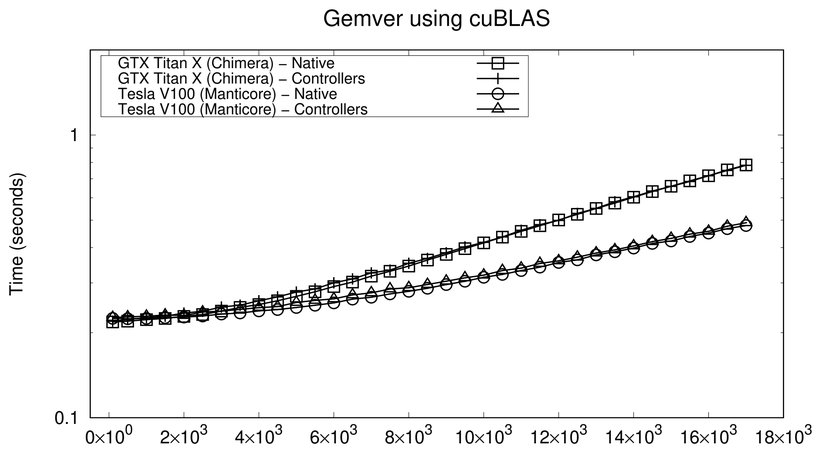
- Extensions to the Controller programming model.
- Abstraction layer for more complex library-calling kernels.
- Manage different coprocessor memory transfers, efficiently and decoupled from kernel and BLAS library calls.
- Extended poly-kernel definition system, supporting contexts, etc.
- Run-time kernel selection policy considering both programmer and different external libraries for each type of device.
- HitCtrlBLAS: Complete new BLAS Controller library (all routines).
- Support: cuBLAS, MAGMA (GPUs, XeonPhi), IntelMKL (CPUs, XeonPhi).

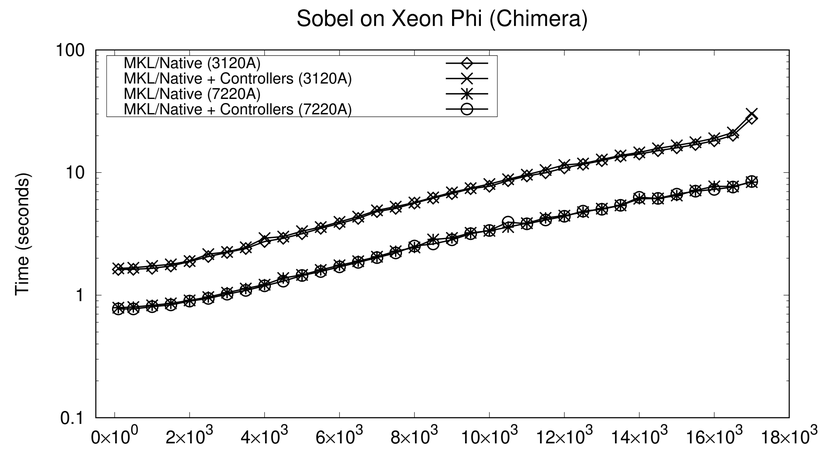


Experimental work

- Case study applications
  - Matrix multiplication sequence: $C = A \times B^n$
    Loop of calls to GEMM, high computational cost.
  - GEMVER: Polybench implementation.
    Sequence of calls to different BLAS routines with low load.
  - Sobel filter for images
    Mixing BLAS routines and programmer defined kernels.
- Platforms
  - GPU Nvidia Titan Black (Kepler).
  - GPU Nvidia Tesla V100 (Volta).
  - Intel XeonPhi coprocessors: KNC 3120A and KNL 7220A.
- Results
  - Minimum overhead comparing with reference codes.







Conclusions

⇒ BLAS portability and interoperability for differnt devices:

(1) Flexible system to use BLAS routines and generic programmer kernels; **(2)** Transparent memory management; **(3)** Hides interface and usage differences of external especilized BLAS libraries; **(4)** Code and performance portability.

⇒ On-going work: **(1)** Extention to multi-device Controller; **(2)** Integrate other kinds of libraries.

References

[1] Eduardo Rodriguez-Gutiez, Ana Moreton-Fernandez, Arturo Gonzalez-Escribano, Diego R. Llanos. Toward a BLAS library truly portable across different accelerator types. The Journal of Supercomputing, 2019 (online-first) DOI: 10.1007/s11227-019-02925-3

This research has been partially supported by MICINN (Spain), the ERDF program of the European Union and Junta de Castilla y Leon: PCAS project (TIN2017-88614-R), CAPAP-H6 (TIN2016-81840-REDT), FEDER Grant VA082P17 (PROPHET Project) and the HPC-EUROPA3 project (European Commission H2020 Research and Innovation). HPC-Europa3 is supported by the European Commission H2020 Research & Innovation GA # 730897