Towards a Heterogeneous data-centric framework for efficient Linear algebra

Petros Anastasiadis†, Georgios Goumas†

†CSLab, School of Electrical and Computer Engineering, National Technical University of Athens, Greece

Introduction

- BLAS(Basic Linear Algebra Subprograms) is extensively used in scientific code.
- The existing highly optimized vendor BLAS libraries target only their architectures.
- BLAS currently requires high programming effort to utilize heterogeneous architectures.

Previous Work on Heterogeneous BLAS

- System/architecture/software **dependant**.
- Is either purely **theoretical** or uses problem-specific **low portability** optimizations.
- Makes serious **assumptions** for the data (e.g. location, availability, tiling method).
- Uses task based workload distribution fit for **specific system** characteristics.

Approach	Coverage	System	Implement.	Perf. Model	Tiling
Qilin[1]	BLAS 1-3	Single-GPU	CUDA+TBB	Empirical	1-dim
Tomov[2]	LAPACK	Single-GPU	CUBLAS	No	Square
Werkhoven[3]	1-dim kernel	Single-GPU	CUDA	Semi-empirical	1-dim
cuBLASXT[4]	BLAS 3	Multi-GPU	CUDA (close)	No	Square
BLASX[5]	BLAS 3	Multi-GPU	CUDA	No	Square

Proposed High Level BLAS Framework

A framework that implements **hybrid BLAS** routines with:

- An **unmodified** backwards-compatible BLAS API.
- The **integration** of existing BLAS libraries for computation.
- Efficient **tiling/splitting** methods for data distribution on heterogeneous components.
- A lightweight **prediction model** ensuring near-optimal hybrid performance.
- A **high level** structure which will allow contributing to the framework easier.

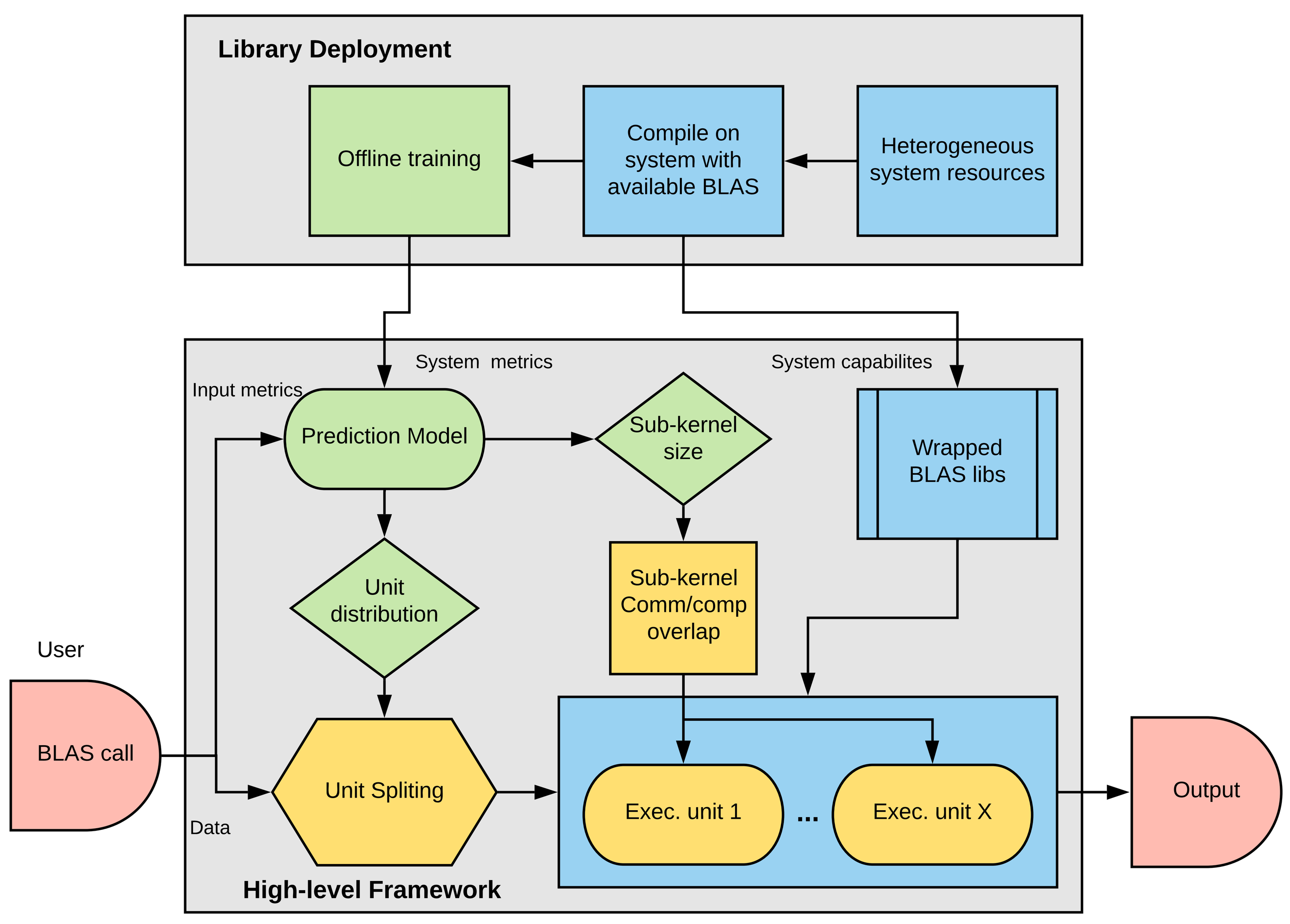

Figure 1: Required components of the proposed Data-centric Framework.

Proposed Prediction Model

We use a **semi-empirical** model approach since BLAS performance characteristics vary **greatly** for different architectures/routines/underlying library optimizations:

- **Execution units** (ex) are **heterogeneous components** which can operate on data.
  - Units are **independent** and support heterogeneous **parallel execution**.
  - Each unit is assigned a **sub-set** of the initial problem called unit **work set** (WS_{ex}).
- **Data Links** (dl) are the available **connections** between units used to transfer data.
- Each work set can be further split in equal **data chunks** called **Sub-kernels** (SK_{ex}).
  - Sub-kernels are used for unit-link wise **comm/comp overlap**.
- In a heterogeneous system with x units the total time of a program is modeled as such:

$$t_{total} = max(t_1(WS_1, SK_1), t_2(WS_2, SK_2), ...t_x(WS_x, SK_x)) \quad (1)$$

$$t_x(WS_x, SK_x) \approx t_{send\_first}(SK_x) + \frac{WS_x}{SK_x} t_{over}(SK_x) + t_{recv\_last}(SK_x) \quad (2)$$

$$t_{over}(SK_x) \approx max(t_{send}(SK_x), t_{exec}(SK_x), t_{recv}(SK_x)) \quad (3)$$

$$t_{send}(SK_x) = t_{dl}(SK_x, mem_x, mem_{src}), t_{recv}(SK_x) = t_{dl}(SK_x, mem_{dest}, mem_x) \quad (4)$$

$$t_{dl}(SK_x, to, from) \approx lat_{dl} + G_{dl} * bytes_{SKx}, G_{dl} = \frac{1}{bw_{dl}} \quad (5)$$

Model Evaluation

- We evaluate the model (1) applied to BLAS **DGEMM** on two systems:
  - System 1: Xeon Gold-5120 (28 cores), $Link \approx 13GB/s$, 1 GTX1060.
  - System 2: Intel i7-4820K (8 cores), $Link \approx 3.2GB/s$, 1 TeslaK40.
- Link lat_{dl}, bw_{dl} (5) estimated with 2-value sampling (LogP equivalent [3]).
- Sub-kernel CPU/GPU **execution time** from (3) estimated with lsq linear regression:
  - **Train** set: ≈ 1000 samples of sizes between 1K and 6K for M,N,K.
- **Validation** sets: 4 different M,N,K classes (fig. 2) with varying tested WS_x, SK_x.

Sys. 1

Val set	S1	S2	S3	S4
Max err(%)	13.75	21.88	2.50	14.32
Min err(%)	-3.69	-3.14	-3.65	-3.86
Mean abs err(%)	4.18	2.14	1.71	4.45

Sys. 2

Val set	S1	S2	S3	S4
Max err(%)	30.12	17.81	8.96	18.46
Min err(%)	-20.69	-6.0	-3.44	-27.3
Mean abs err(%)	6.1	3.42	1.91	5.64

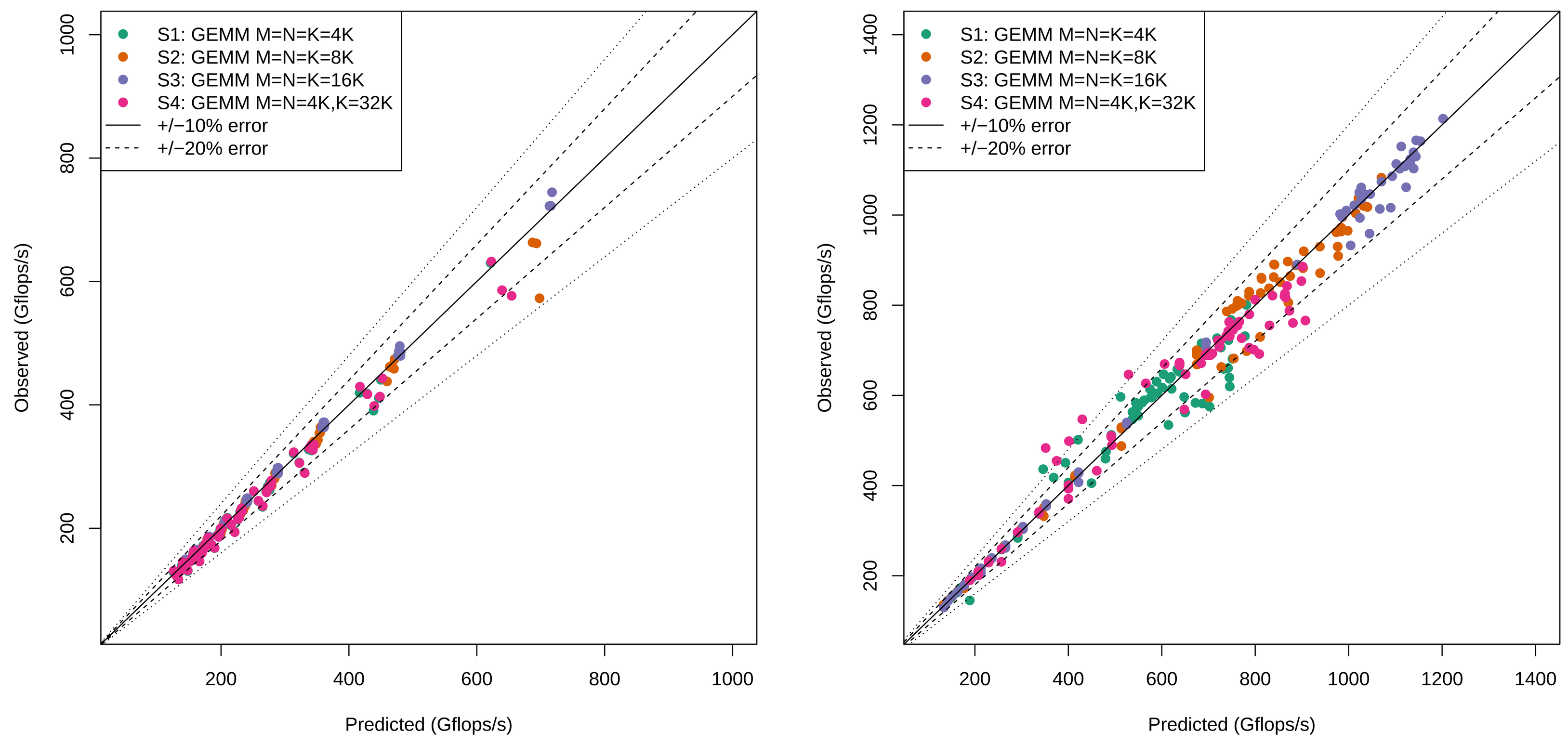

Figure 2: Observed vs Predicted values for the Sub-Kernel model for the execution time of GEMM. The model uses two units (unit 1 = CPU, unit 2 = GPU), with the assumption that the input/output resides on host ($SK_1 = 0$).

- **Low** mean percentile error for both systems, acceptable min/max errors.
- Min/Max errors appear in low-performance areas -> less important for optimization.
- Predictions for sys. 2 scattered since low bw_{dl} increases the performance impact of SK_2.

Preliminary Results

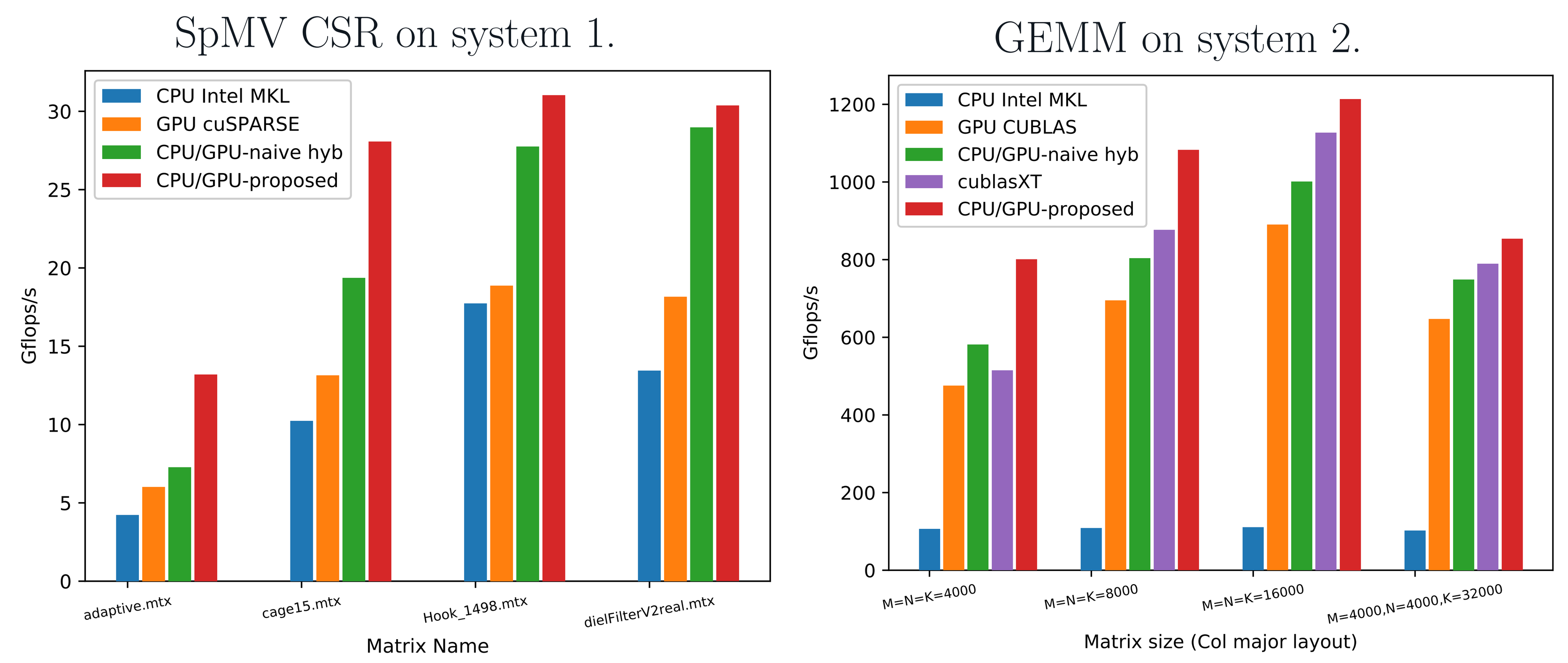

Figure 3: Performance comparison with vendor libraries. The CPU/GPU-Naive version is a wrapped parallel execution of vendor routines with the WS_x predicted with our model without sub-kernel tiling ($SK_x = 0$). The CPU/GPU-subkernel version uses the same wrapped vendor routines for execution and CUDA streams for data tiling and comm/comp overlap, with WS_x, SK_x estimated with bound L-BFGS on eq. (1).

Future Remarks

- Our **model** provides good insight on heterogeneous execution time.
  - Can be used for efficiently predicting WS_x, SK_x with curve fitting methods.
  - Requires empirical tuning and its accuracy depends greatly on the $t_{over}(SK_x)$ formula.
- Data tiling **implementation** still needs optimization/further research.
- BLAS Wrapper must be able to support more heterogeneous systems/paradigms.

References

[1] Chi-Keung Luk, Sunpyo Hong, and Hyesoon Kim. "Qilin: Exploiting Parallelism on Heterogeneous Multiprocessors with Adaptive Mapping". In: *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. 2009.

[2] Stanimire Tomov, Jack Dongarra, and Marc Baboulin. "Towards dense linear algebra for hybrid GPU accelerated manycore systems". In: *Parallel Computing* 36.5 (2010).

[3] B. v. Werkhoven et al. "Performance Models for CPU-GPU Data Transfers". In: *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. May 2014.

[4] Nvidia. *CublasXT*.

[5] Linnan Wang et al. "BLASX: A High Performance Level-3 BLAS Library for Heterogeneous Multi-GPU Computing". In: *Proceedings of the 2016 International Conference on Supercomputing*. ICS '16.