# MASCOT: Fast and Highly Scalable SVM Cross-validation using GPUs and SSDs

Zeyi Wen, Rui Zhang, Kotagiri Ramamohanarao,
Jianzhong Qi, Kerry Taylor[†]

*Department of Computing and Information Systems*
*The University of Melbourne, Australia*

[†]The Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia

# Overview of the SVM cross-validation

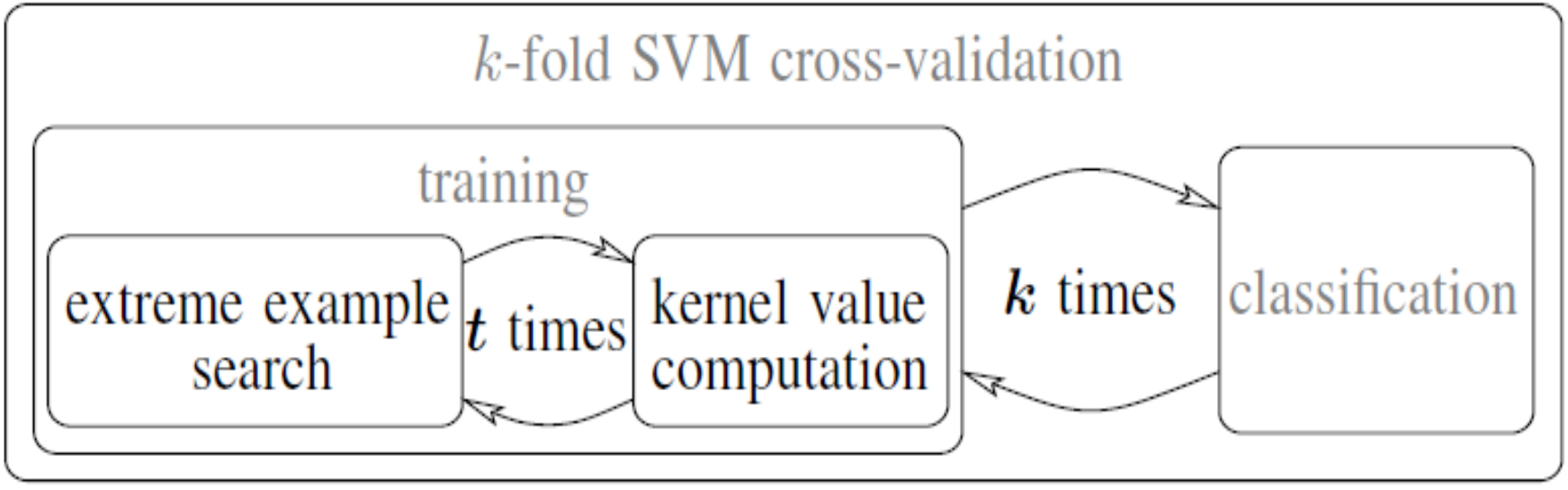



Fig. 1: $k$-fold SVM cross-validation

**SVM cross-validation is expensive, time complexity of $\mathcal{O}(ktnd)$!**

# *Limitation of the existing work & our goal*

- CPU based SVM implementation is slow because of the expensive computation in the training (**inefficient**).
- GPU based SVM implementation requires storing the whole training dataset in the GPU global memory which is small (**not scalable**).

Our goal is to achieve high **scalability** and high **efficiency**.

# *Key ideas*

- Kernel matrix precomputation and reusage
  - Benefits: avoid repeated computation and holding the dataset
  - Use GPU memory for caching kernel values
  - Store the matrix to the CPU memory extended by SSDs
- Optimise the search algorithm
- The improved time complexity is $\mathcal{O}(ktn)$.

# Overview of our solution

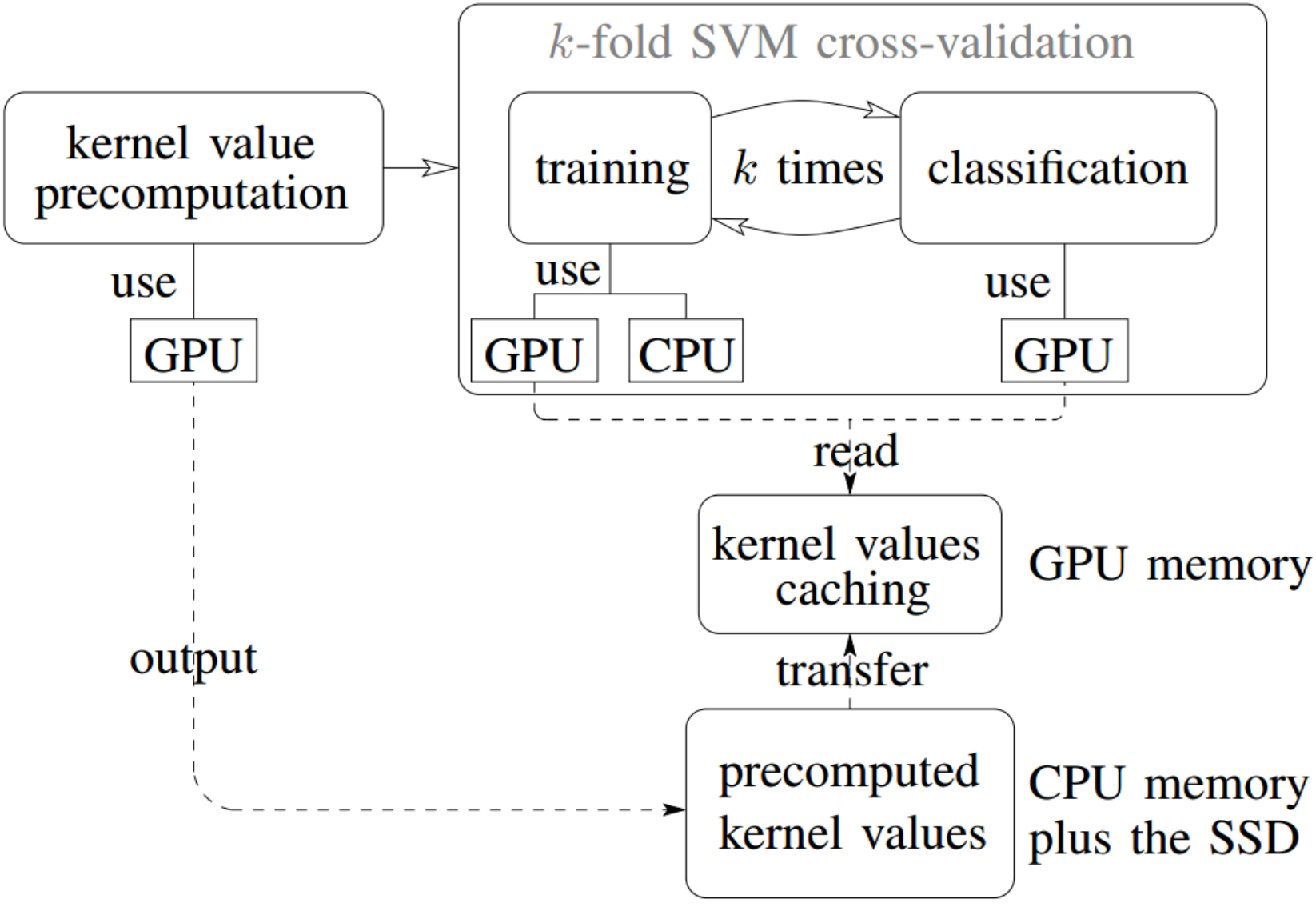



Fig. 3: The MASCOT scheme

# *Storing kernel values*

Store kernel values to the CPU memory extended by the SSD.

- Use “Zero-copy” for the part in the CPU memory
- Read the SSD in parallel.

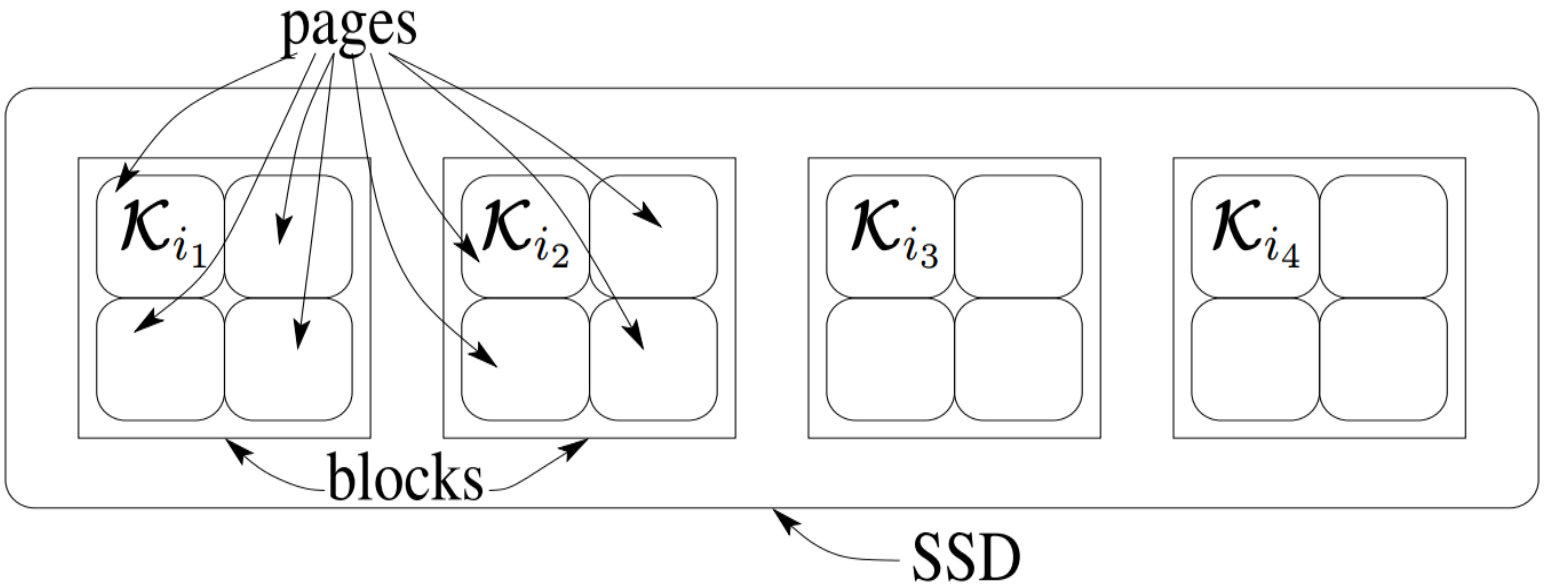



Fig. 4: Storing the row $\mathcal{K}_i$ to an SSD

# *Caching strategy*

We analyse the SVM training, and discover that the kernel value's access pattern is similar to sequentially scan all the kernel values for multiple times.

The general purpose LRU strategy is not suited for SVM training. Our strategy essentially caches a fixed part of the kernel matrix.

# The search algorithm

* Make use of registers to reduce the accesses to the shared memory.
* Terminate threads when their tasks are completed.

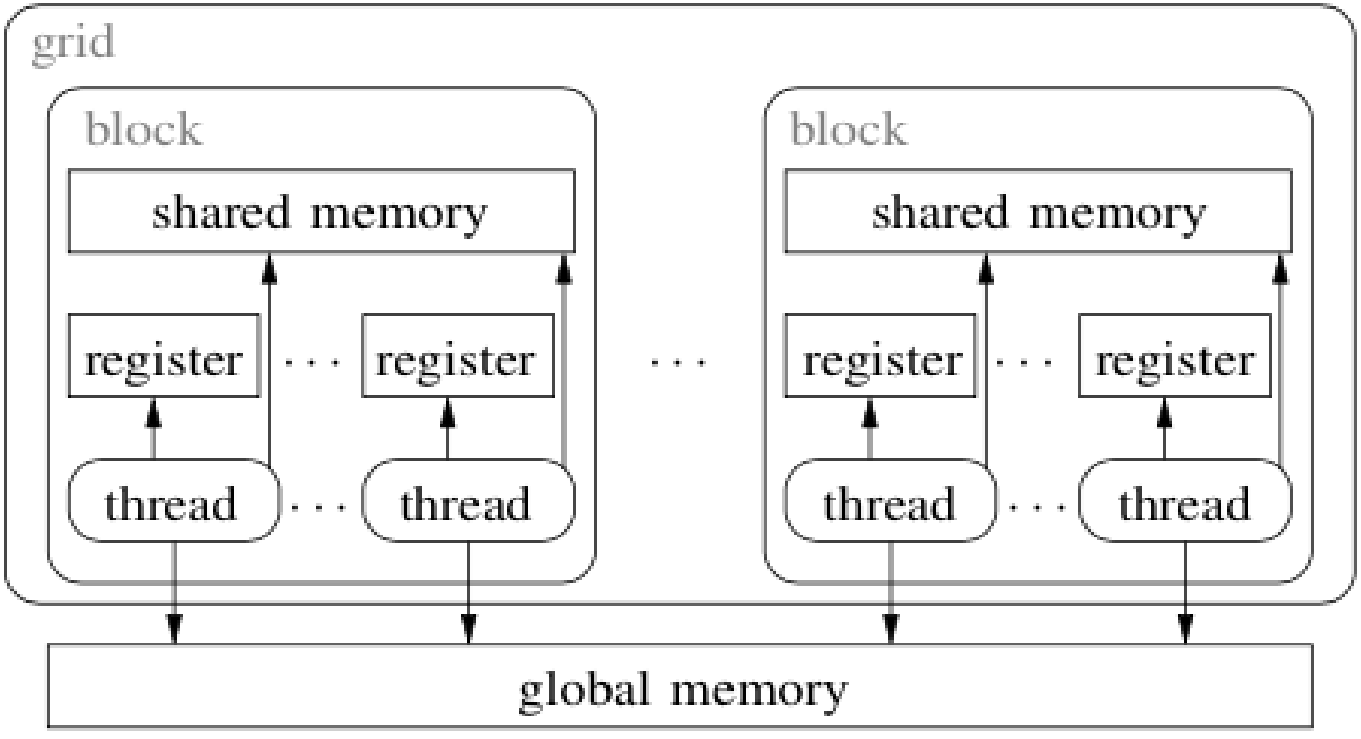



Fig. 2: Threads and memory on a GPU

# *Experimental settings*

- We conducted the experiments on a computer running Linux with a Xeon E5-2643 CPU, 16GB CPU memory, a 240GB SSD and a GTX460 GPU with 768MB memory.
- We performed 10-fold cross-validation.
- Baselines are LibSVM in C++ and gSVM in CUDA-C.

# *Experiments: datasets and parameters*

Table : Datasets and kernel parameters

| dataset | cardinality | dimension | $C$ | $\gamma$ |
|---|---|---|---|---|
| Adult | 32,561 | 123 | 100 | 0.5 |
| Epsilon | 120,000 | 2,000 | 0.01 | 1 |
| Gisette | 6,000 | 5,000 | 100 | 0.382 |
| MNIST | 60,000 | 780 | 10 | 0.125 |
| RCV1 | 20,242 | 47,236 | 100 | 0.125 |
| Webdata | 49,749 | 300 | 64 | 7.8125 |

# *Effectiveness of the caching strategy*

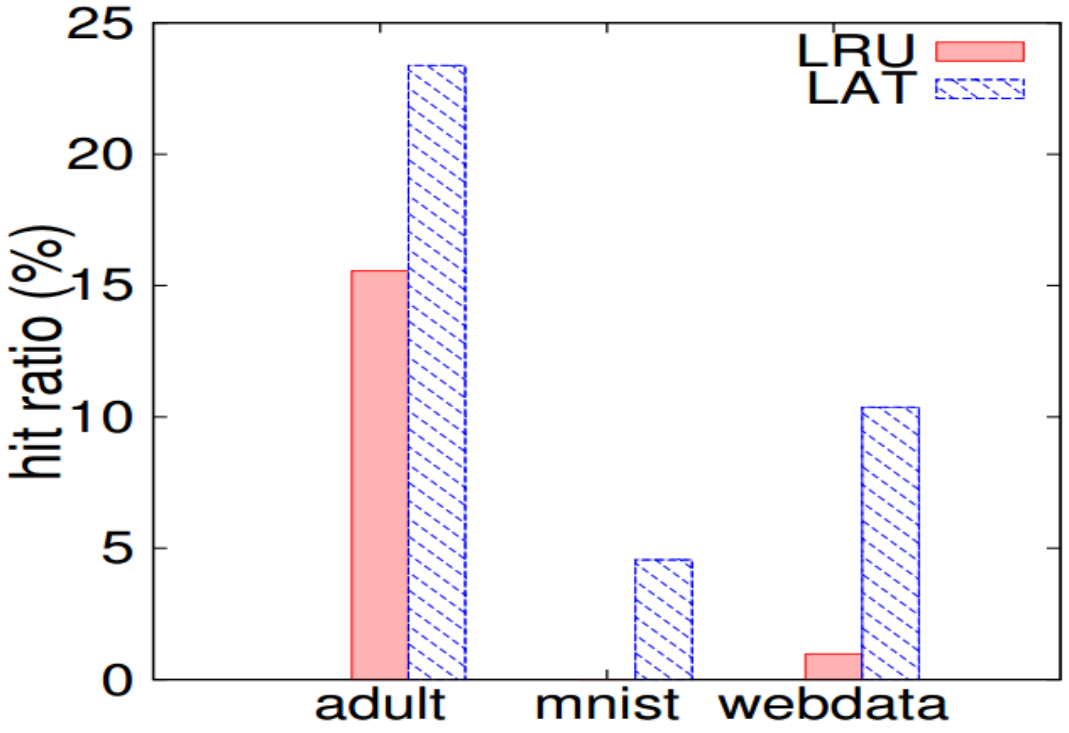



Fig. 6: Caching strategies

# *Efficiency of reading kernel values in parallel*

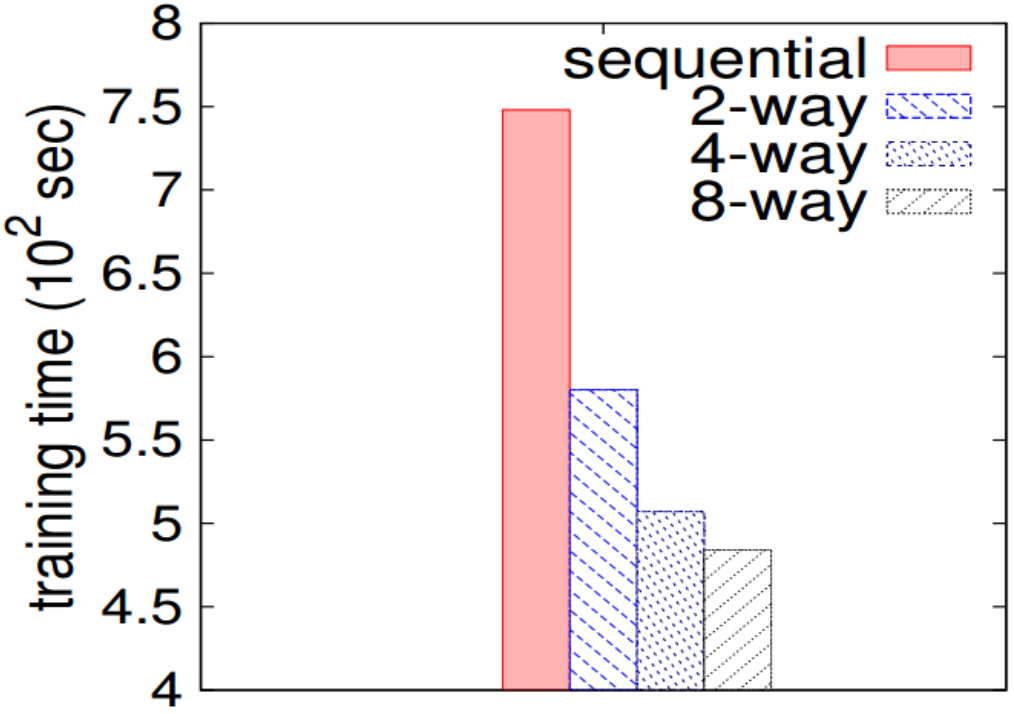
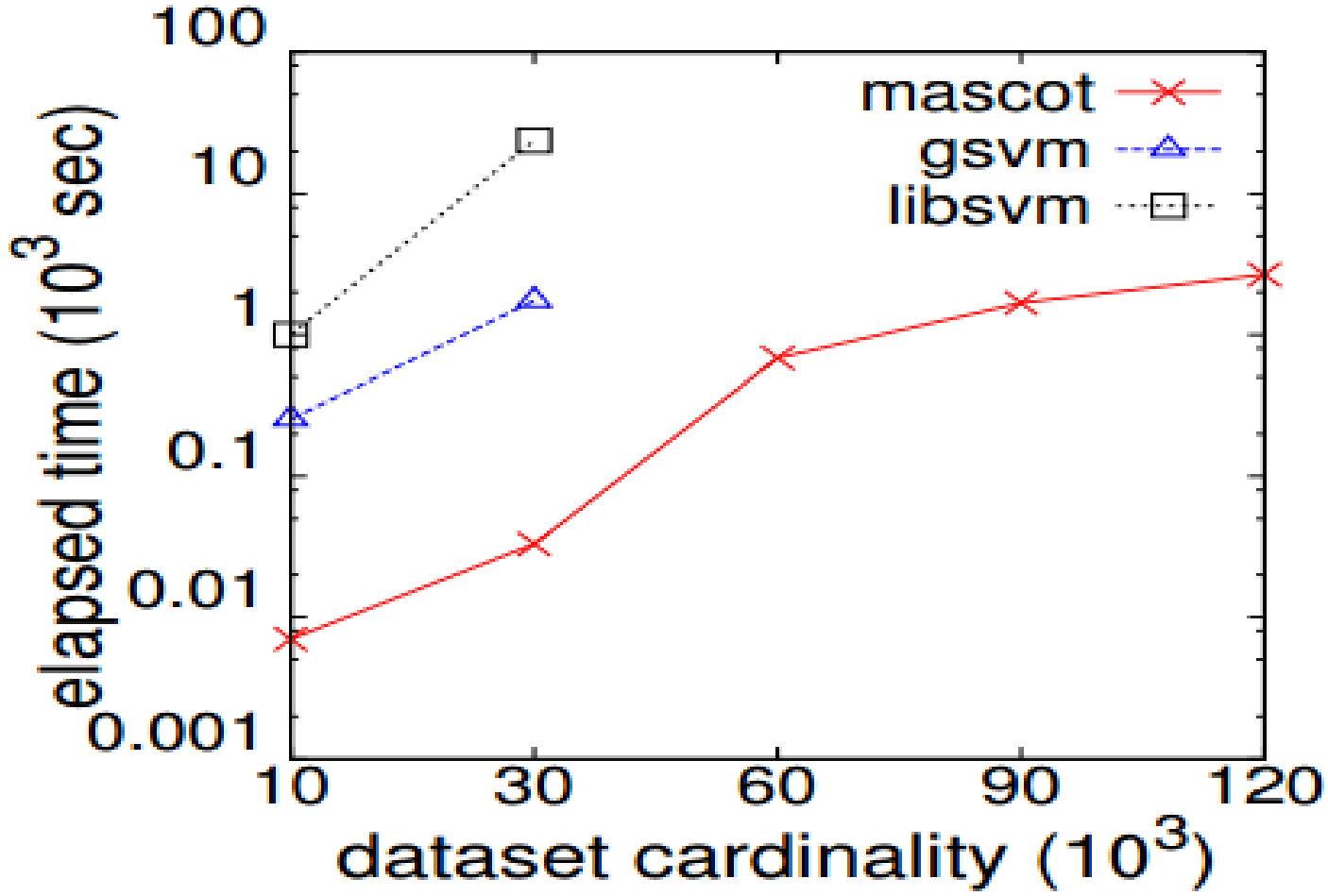


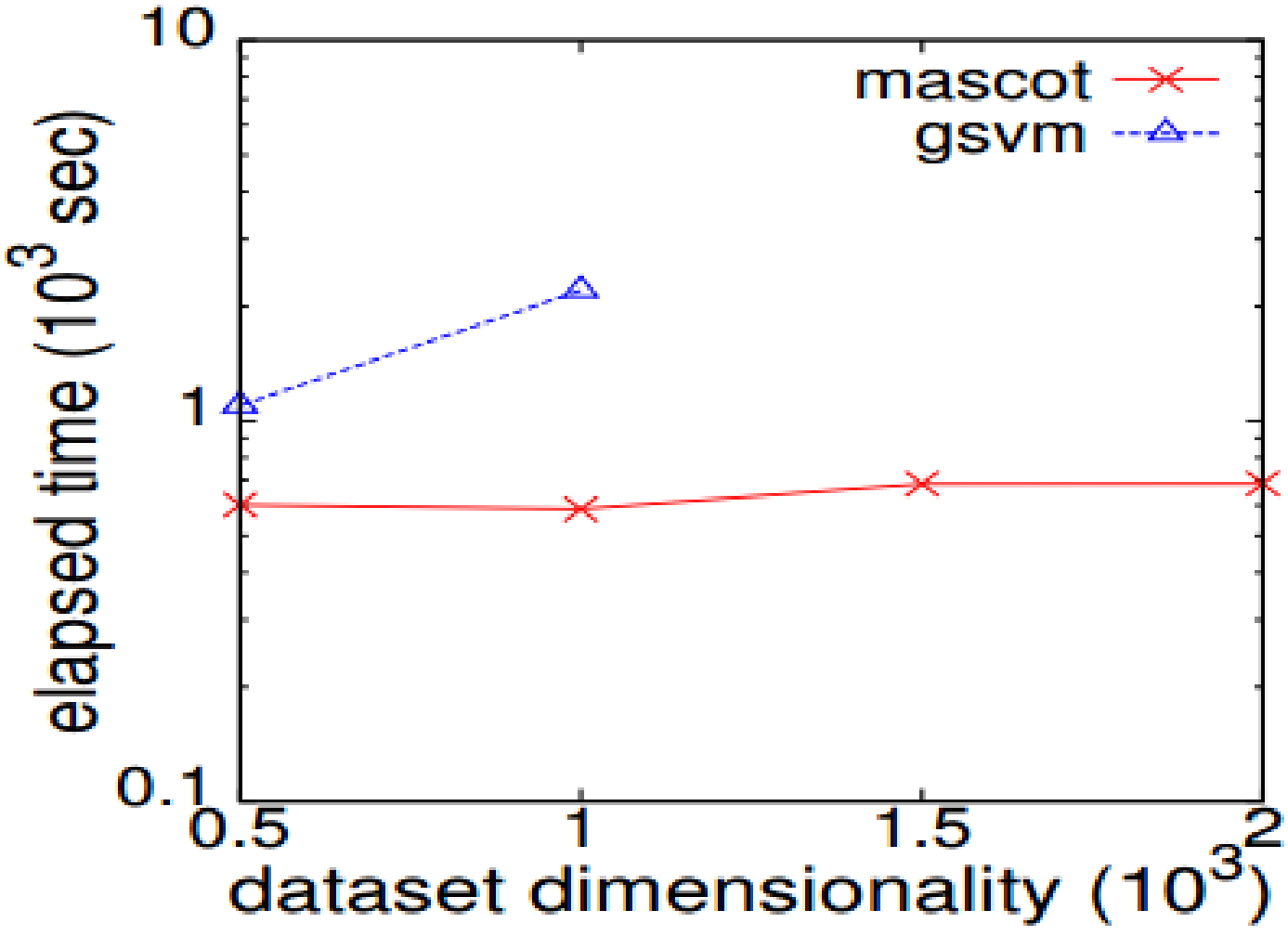

Fig. 7: Parallel kernel value read

# *Overall scalability*




(a) Cardinality (Epsilon)

# *Overall scalability*




(c) Dim. (Epsilon)

# Overall efficiency

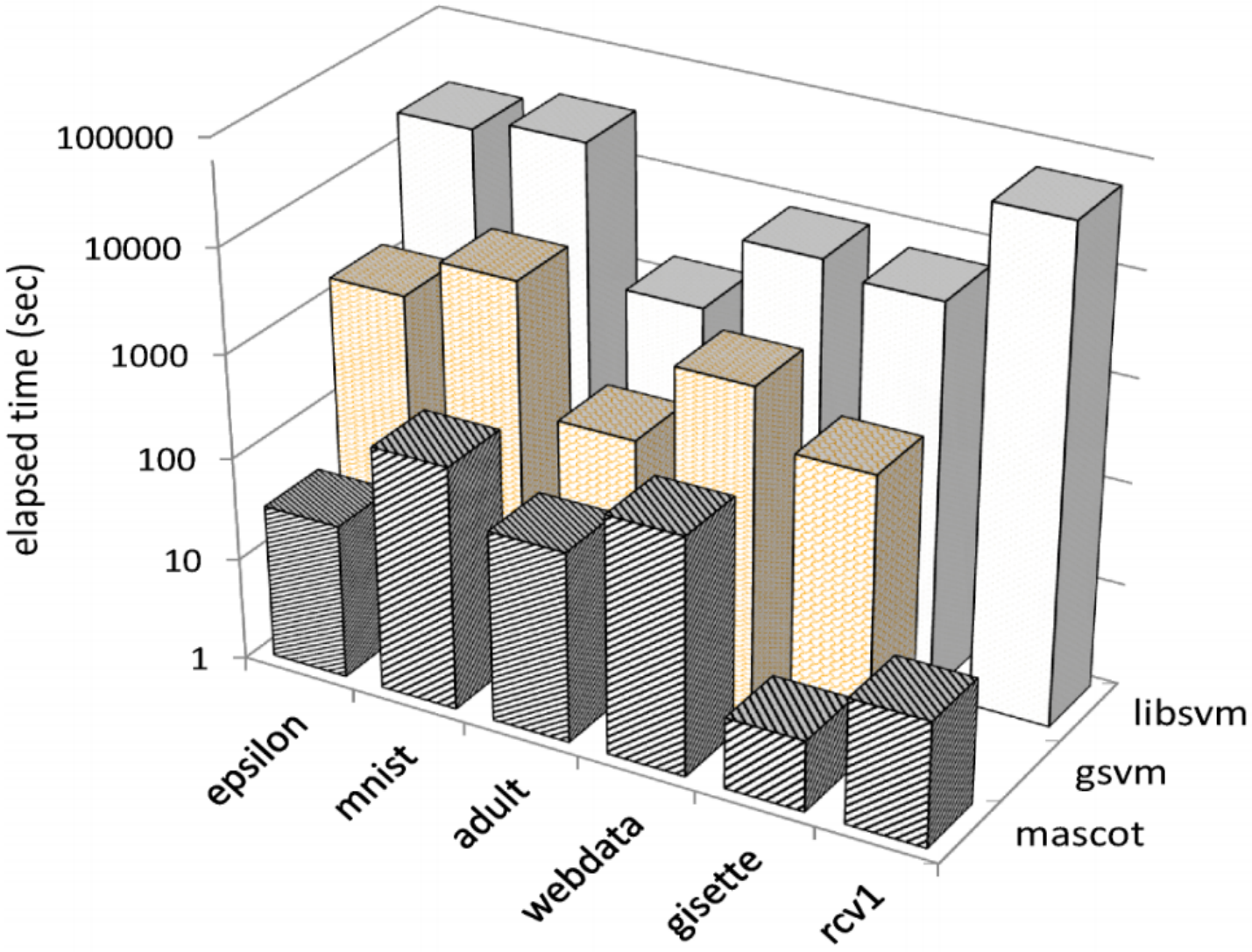



Fig. 10: Efficiency

# *Conclusion*

- We proposed a scheme for scalable and fast SVM cross-validation by exploiting the high performance of GPUs and SSDs.
- Precomputation, caching, optimised search algorithm, design a parallel kernel value read algorithm.

# *Source code*

Source code is available online.


Zeyi Wen: zeyiw@student.unimelb.edu.au