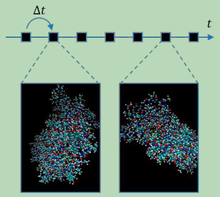
A HIGHLY SCALABLE KERNEL BASED CLUSTERING ALGORITHM FOR MOLECULAR DYNAMICS

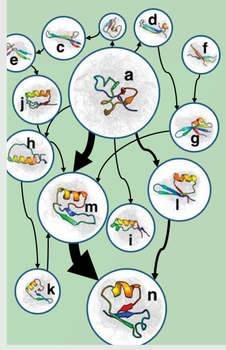
M. J. Ferrarotti, S. Decherchi and W. Rocchia
CONCEPT Lab, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genova, Italy
ISTITUTO ITALIANO DI TECNOLOGIA

CLUSTERING MD TRAJECTORIES

A BIG DATASET IN HIGHLY DIMENSIONAL SPACE



- **MD trajectory:** discretized **time evolution of the atomic positions** for an entire molecular system.
- Up to $\mathbf{10^{10}}$ **frames** (10μs simulation, 2fs timestep)
- Each frame: $\sim\mathbf{10^5}$ **atom coordinates** in 3D space



GETTING VALUE OUT OF DATA

- Biomolecular processes of interest evolve through a series of metastable states. *How do we define those states in a rigorous unbiased way? Can we build reliable kinetic models?*
- Clustering is the core tool in probabilistic coarse grained model such as **Markov State Models** [1].
- Clustering by itself can be used to extract human-interpretable mesostates to describe a process [2].

A NOVEL ALGORITHM SUITED FOR MD: MINIBATCH DISTRIBUTED KERNEL K-MEANS

DESIGN GUIDELINES

- **Avoid featurization procedures:** they are expensive and inaccurate, our algorithm is based on **Kernel K-means** which **does not require an explicit feature space**. Indeed the whole algorithm requires just a distance metric being expressed in terms of a kernel matrix with the following form,

$$K_{i,j} = e^{-\gamma\, d^2(x_i,x_j)}\,;\; d^2 = \text{aligned RMSD}^2(x_i,x_j) = \frac{1}{N_{atoms}}\sum_{k=1}^{N_{atoms}}\left\|x_{i,k}-x_{j,k}\right\|^2$$

- **Target HPC facilities:** being inspired by the work of Zhang, Rudnicky [3] we devised a **fully distributed algorithm** based on the following quantities,

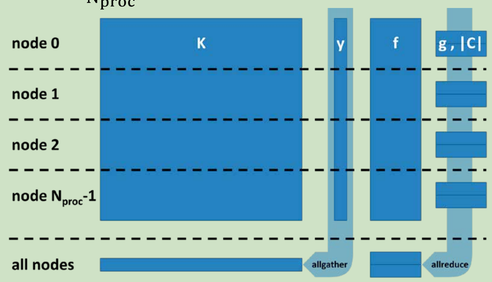
$$f(x_i, C) = -\frac{2}{|C|}\sum_{x_j\in C} K(x_i,x_j);\; g(C) = \frac{1}{|C|^2}\sum_{x_i\in C}\sum_{x_j\in C} K(x_i,x_j)$$

- **Think big:** to scale on large datasets we **divide the data into mini-batches** which are presented to the algorithm one after the other [4].

e.g. 10^6 frames › 16 nodes, 32 GB /node › 4 minibatches (dobule precision), 3 minibatches (single precision)

DISTRIBUTION STRATEGY

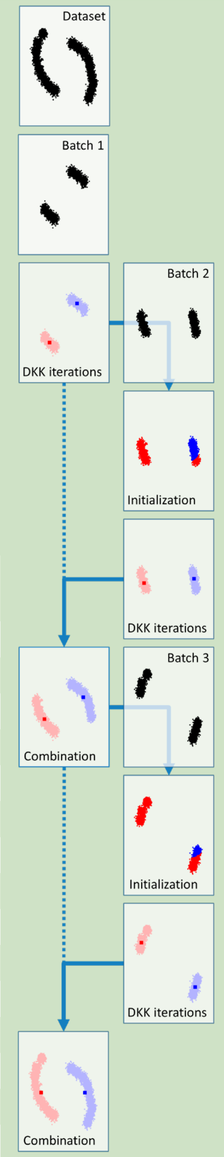
- K, f matrices as well as labels y are fully distributed.
- Each node holds a partial copy of g and |C| .
- Each node deals with $\frac{N^2}{N_{proc}}$ kernel elements.
- $2N_c + \frac{N}{N_{proc}}$ all-to-all communications per step.



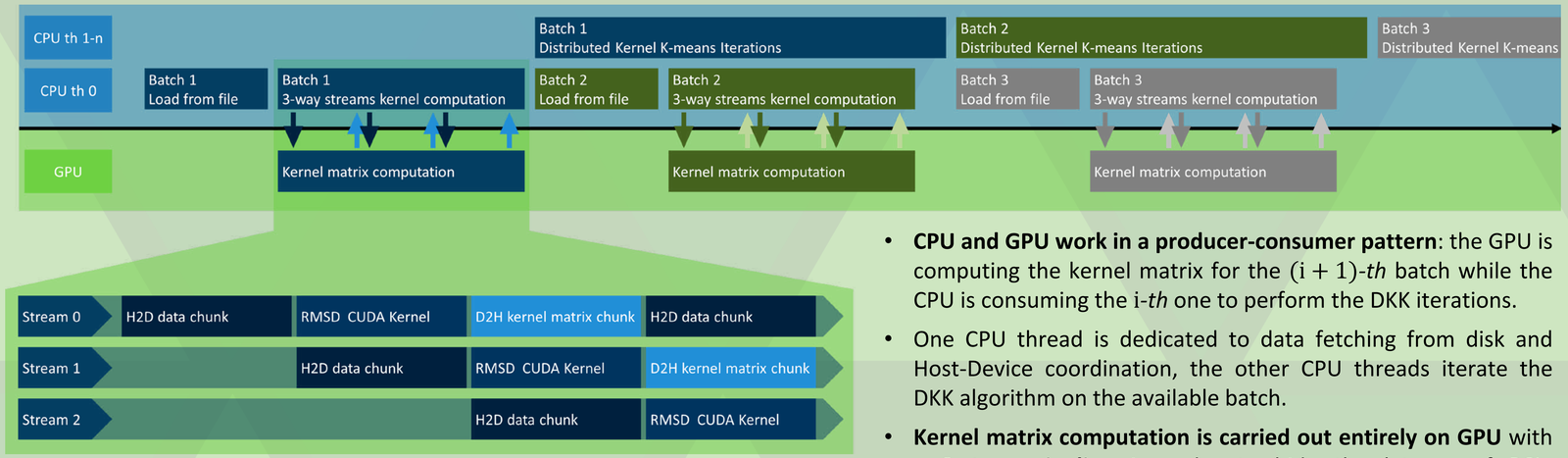
ALGORITHM PSEUDOCODE

D: dataset, B: batch, N: batch size, K: kernel matrix, m: medoids, y: labels

for i=1 to |D|/N **do**
  B ← N frames uniformly sampled from D
  K ← compute distributed kernel matrix
  if (i==1) **then** m_i ← kernelized k-means++ init
  y ← label according to nearest neighbor medoid
  while not converged **do**
    compute local f, g, |C| based on K, y
    allreduce g, |C|
    for every sample: $y_j \leftarrow \text{argmin}_c\,(\, f(x_j,C) + g(C)\,)$
    allgather updated y on each node
  end do
  m_{i+1} ← find mini-batch medoids
  $m_{i+1} \leftarrow a \cdot m_i + (1\text{-}a) \cdot m_{i+1}$; $a=|C|_i/(|C|_i+|C|_{i+1})$
end for



GPU ACCELERATION



- **CPU and GPU work in a producer-consumer pattern**: the GPU is computing the kernel matrix for the $(i+1)$-*th* batch while the CPU is consuming the i-*th* one to perform the DKK iterations.
- One CPU thread is dedicated to data fetching from disk and Host-Device coordination, the other CPU threads iterate the DKK algorithm on the available batch.
- **Kernel matrix computation is carried out entirely on GPU** with a **3-stage pipeline** in order to hide the latency of PCIe communications.

RMSD CUDA KERNEL

- Among the possible algorithms to compute **best-aligned RMSD** we picked the **quaternion based method proposed by Theobald** [5].
- It is **fast and stable** (can be implemented successfully in **mixed single/double precision** fashion).
- To the best of our knowledge the only publicly available GPU implementation of such algorithm is the one introduced by Gil and Guallar [6] in the python package pyRMSD. Such an implementation shows good speedups but still there is **room for improvements**.
- It's our plan to work on the data layout in memory to improve GPU performances (in particular **AoS to SoA transition will be explored to allow coalesced memory access**).

CONCLUSIONS AND OUTLOOKS

- We present a distributed kernel k-means based **algorithm to perform large scale clustering on MD trajectories**. The algorithm is **fully implemented in C++, distributed over MPI**, modular and extensible.
- Single node performances are not sufficient for real-world scenarios without GPU acceleration. **The offload strategy presented here is ideally paired with a self-tuning procedure in order to maximally squeeze the hardware of modern HPC GPU-endowed facilities.**
- The exploitation of approximate techniques where **sparsity on the kernel matrix is induced** by randomly selecting a subset of samples as representation basis [7], **aiming at unprecedented scales** (10^9 - 10^10 frames), which are compatible with the huge amount of simulative data produced in the foreseeable future.

REFERENCES

[1] Voelz, Vincent A., et al. *Journal of the American Chemical Society* 132.5: 1526-1528 (2010).
[2] Decherchi, Sergio, et al. *Nature communications* 6 (2015).
[3] Zhang, Rong, et al. *Pattern Recognition, 16th International Conference on*. Vol. 4. IEEE (2002).
[4] Sculley, David. *Proceedings of the 19th international conference on WWW*. ACM,(2010).
[5] Theobald, Douglas L. *Acta Crystallographica Section A* 61.4: 478-480 (2005): .
[6] Gil, Víctor A., and Víctor Guallar. *Bioinformatics*: btt402 (2013).
[7] Chitta, Radha, et al. *Proceedings of the 17th ACM SIGKDD*. ACM (2011).

ACKNOWLEDGEMENTS

- **CompuNet** – *Computational unit for Drug Discovery and Development at IIT* – for the support
- **CINECA** – *Italian computing center* – for the HPC facilities
- **PRACE** – *Partnership for advanced computing in Europe* – for the HPC facilities
- **AIRC** – *Italian association for cancer research* – for funding (MFAG project 11899)