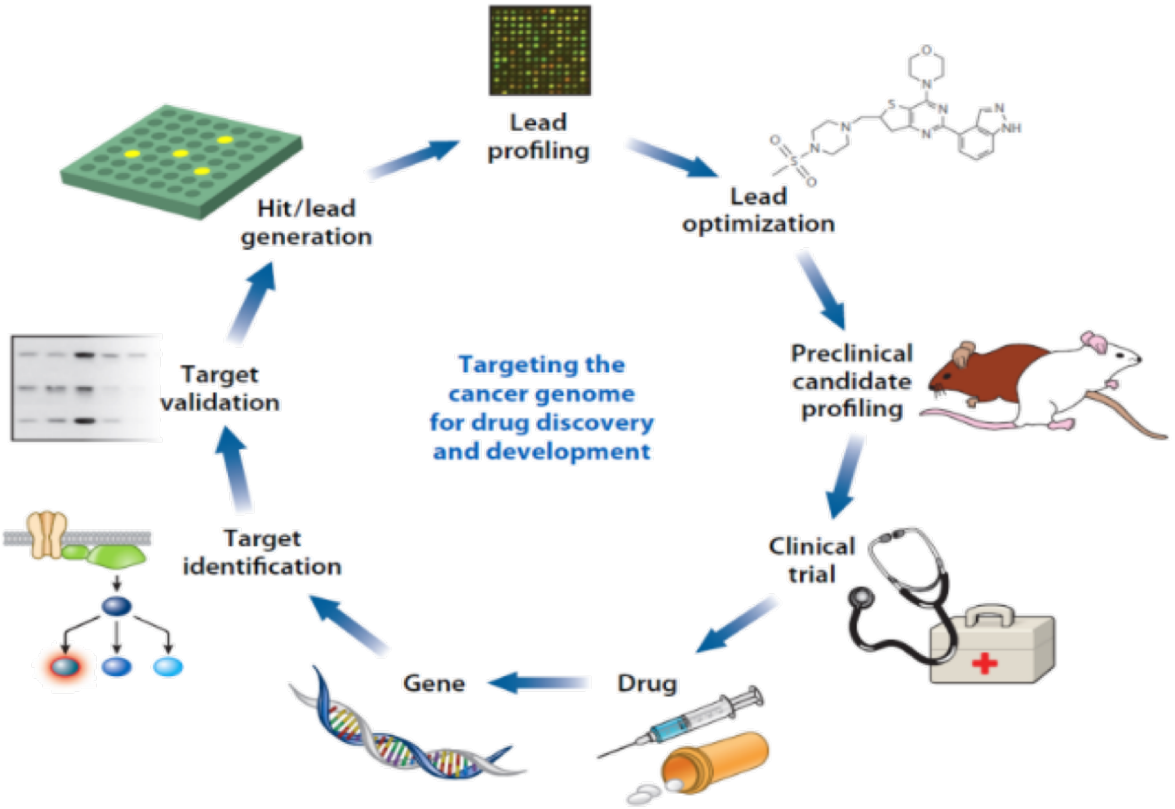
EXASCALE MATRIX FACTORIZATION:
MACHINE LEARNING ON SUPERCOMPUTERS TO FIND NEW DRUGS

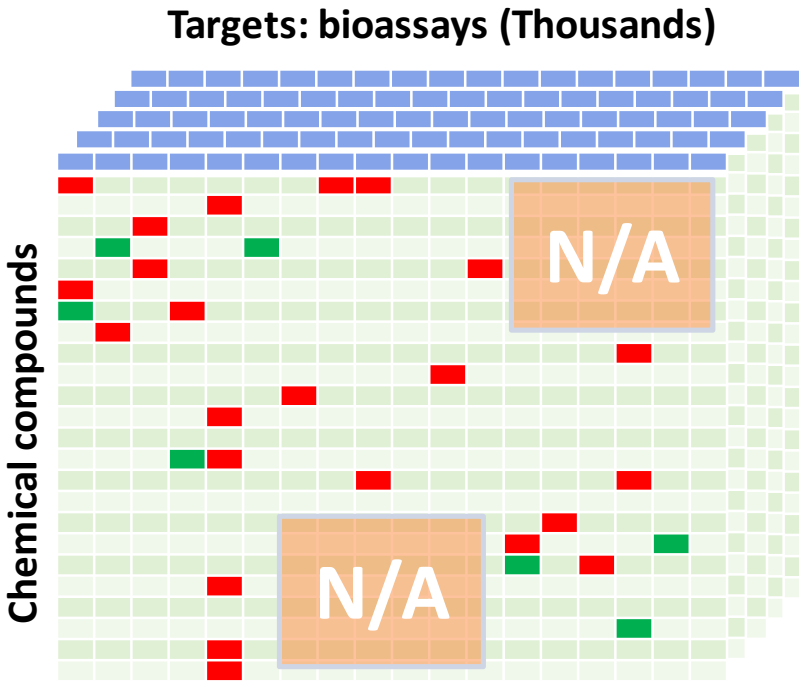
TOM VANDER AA
EXASCIENCE LIFE LAB

DRUG DEVELOPMENT IS TOO EXPENSIVE



COMPOUND ACTIVITY PREDICTION

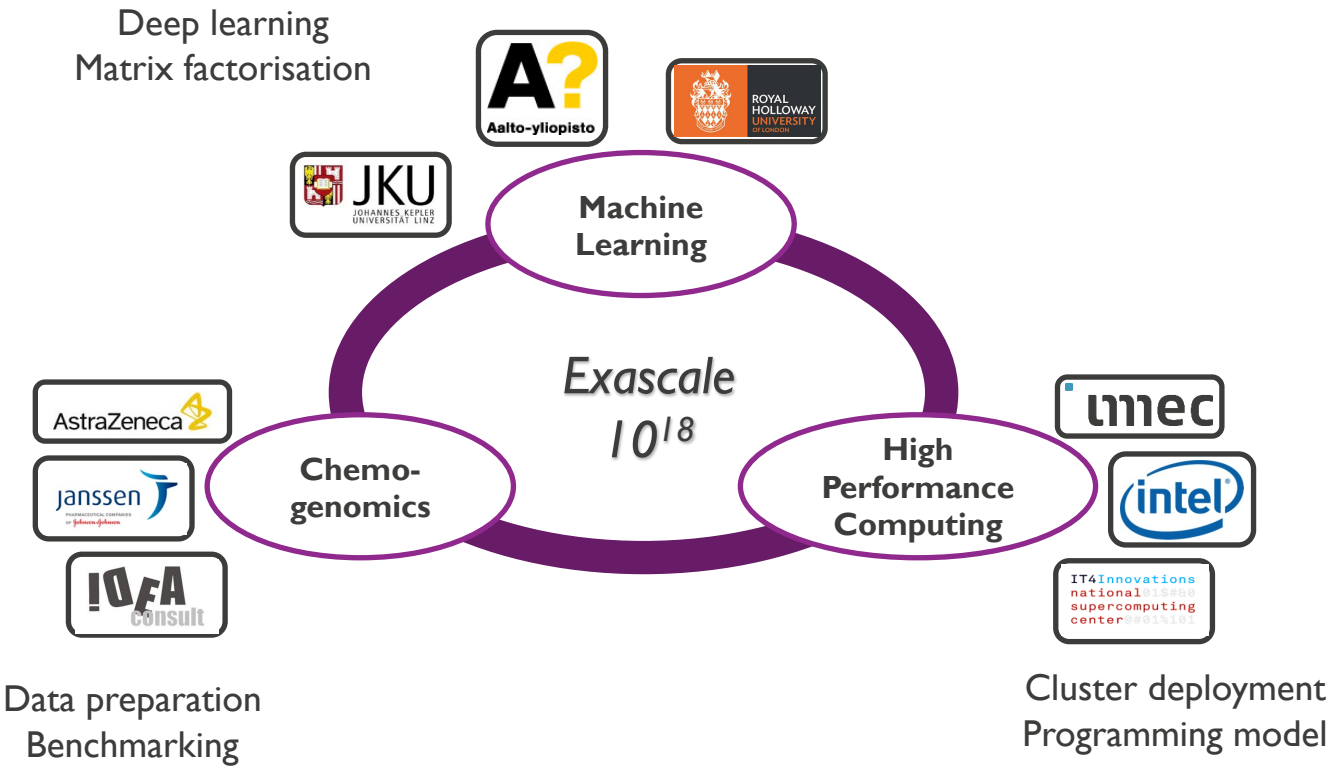
- Predict
  - compound activity on
  - protein target
  - aka chemogenomics
- Like
  - Netflix: users rating movies
  - Amazon: users rating books



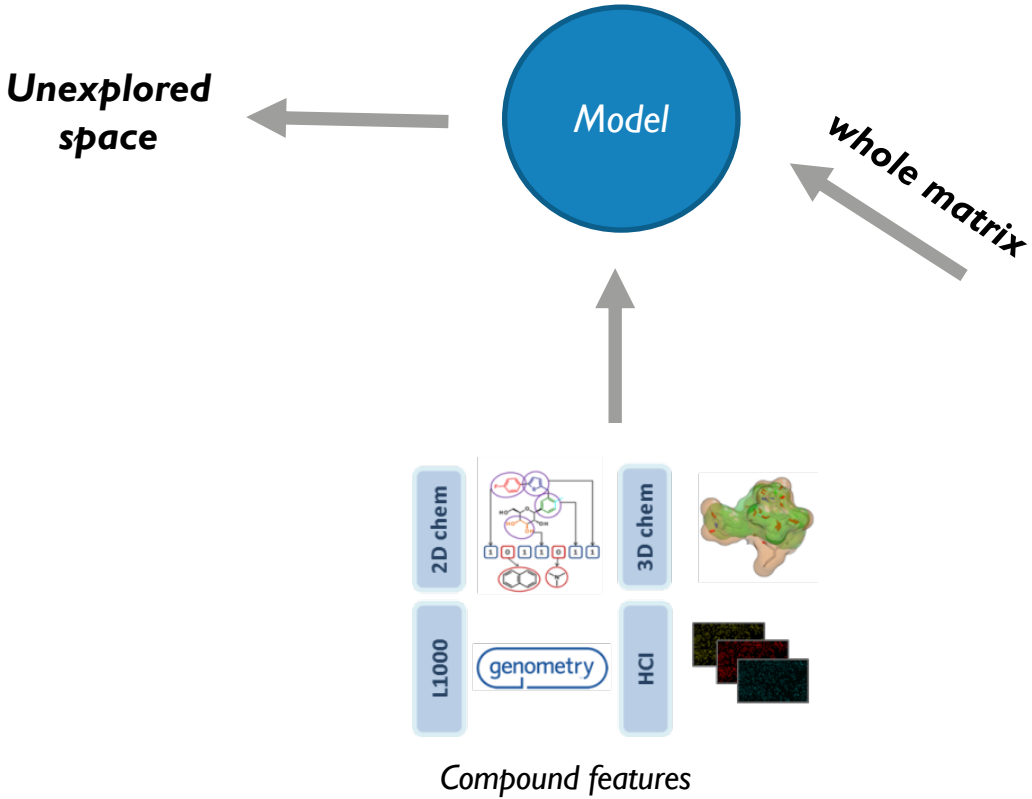
NETFLIX amazon

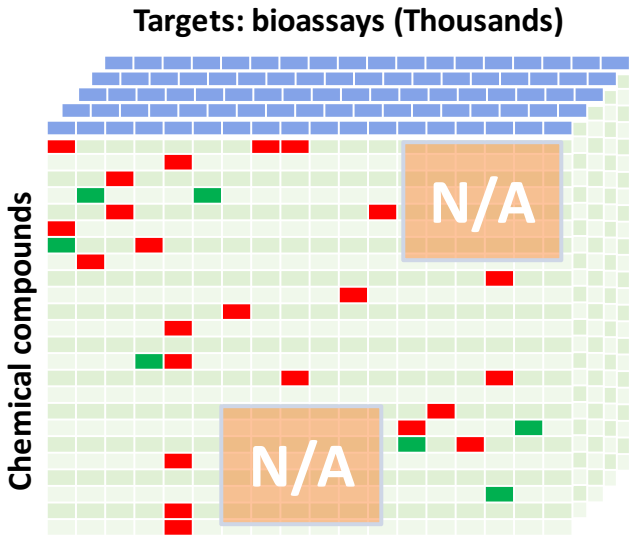
EXCAPE PROJECT

EXASCALE COMPOUND ACTIVITY PREDICTION ENGINES

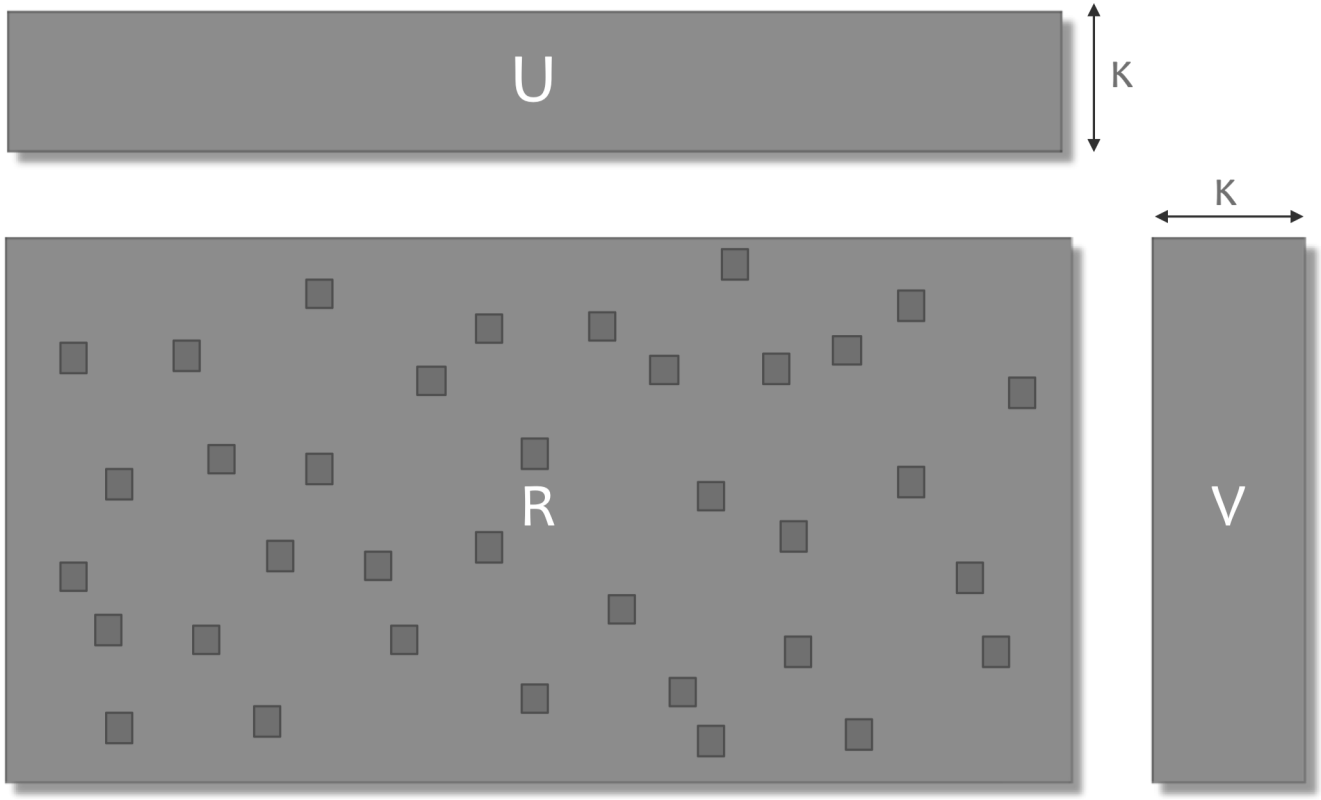


MULTI TARGET CHEMOGENOMICS





BPMF := LOW-RANK MATRIX FACTORIZATION

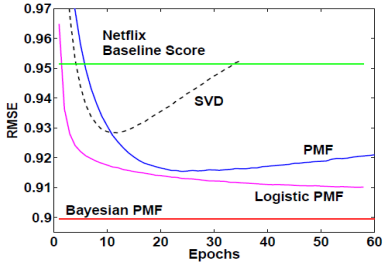


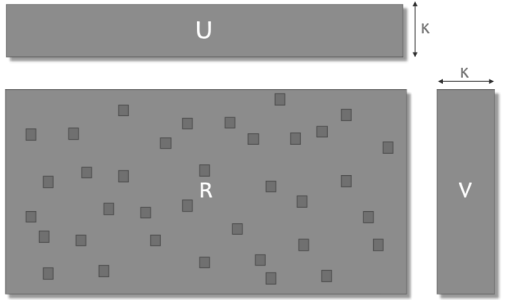
amazon

NETFLIX

FROM SIMPLE, PROMISING BUT SLOW …

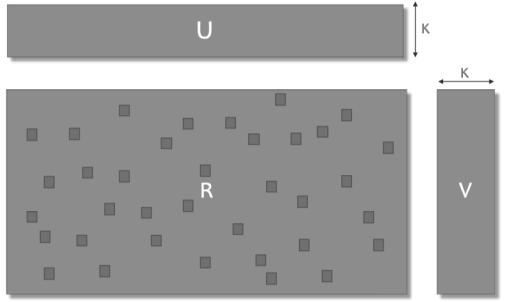
- BPMF is simple
  - 25 lines of Julia code
  - 35 lines of Eigen C++ code
- BPMF predicts well
  - Bayesian → confidence interval
- BPMF is slow
  - Sampling based
  - Julia prototype: 15 days / run

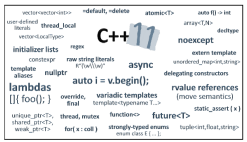




… TO FAST AND COMPLEX

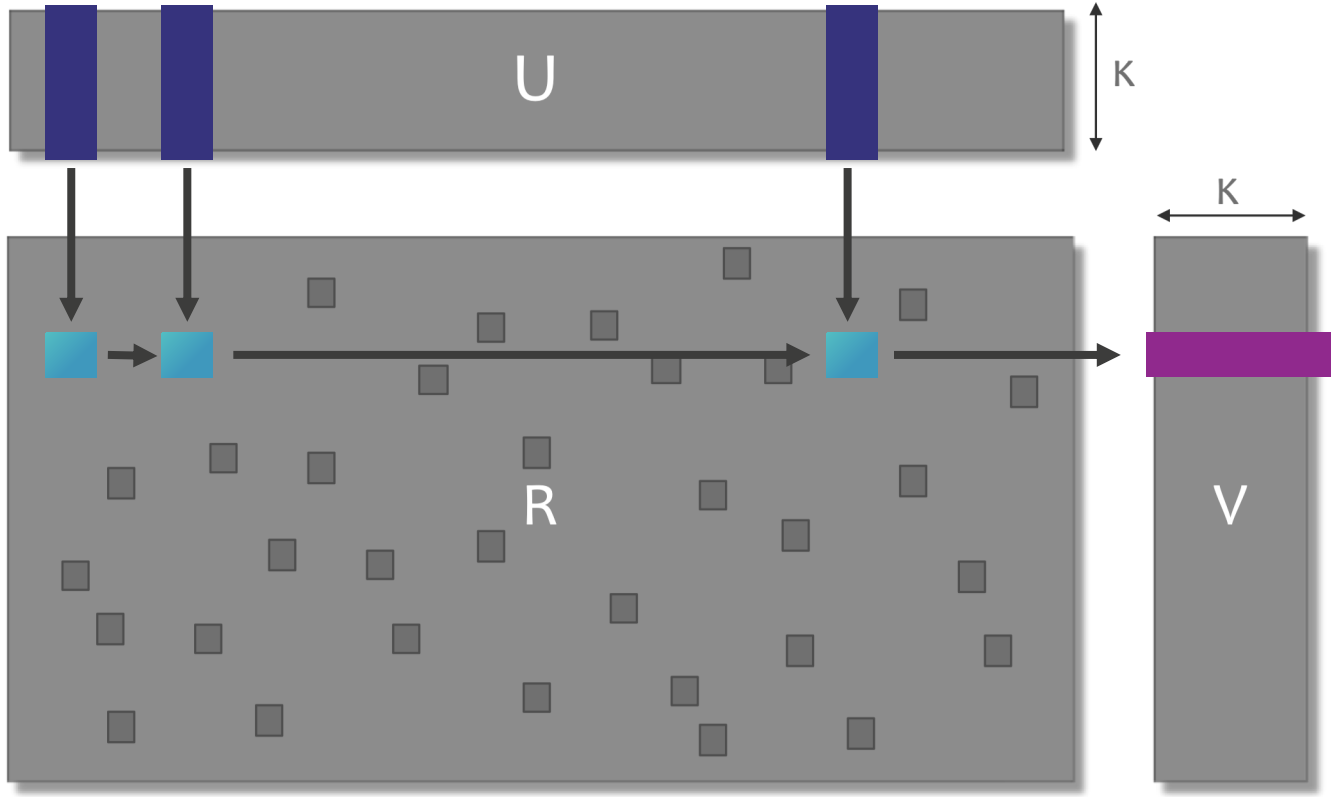
- Many Optimizations
  - Algorithmic
  - Memory Hierarchy
- Multi-core
  - Load Balancing
  - using OpenMP and TBB
- Multi-node
  - Asynchronous communication
  - using MPI and GASPI





BPMF COMMUNICATION AND COMPUTATION

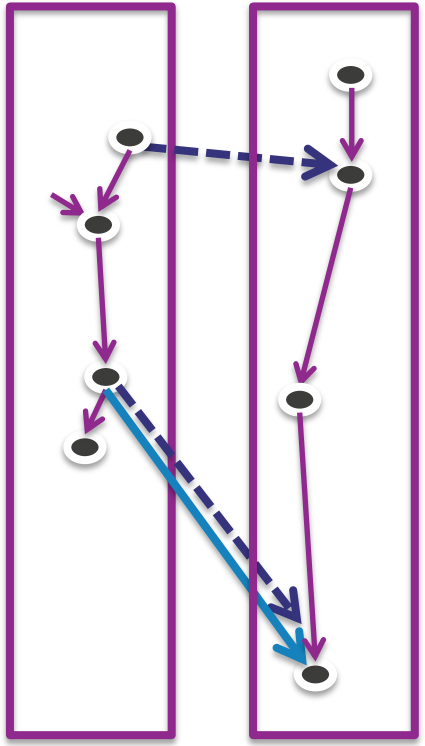
IS DETERMINED BY STRUCTURE OF R



GASPI IN A NUTSHELL

- PGAS API - designed to be
  - Multithreaded
  - Global asynchronous dataflow
  - Interoperability with MPI

- GASPI Specification (gaspi.de)
- GPI-2 Implementation (gpi-site.com)



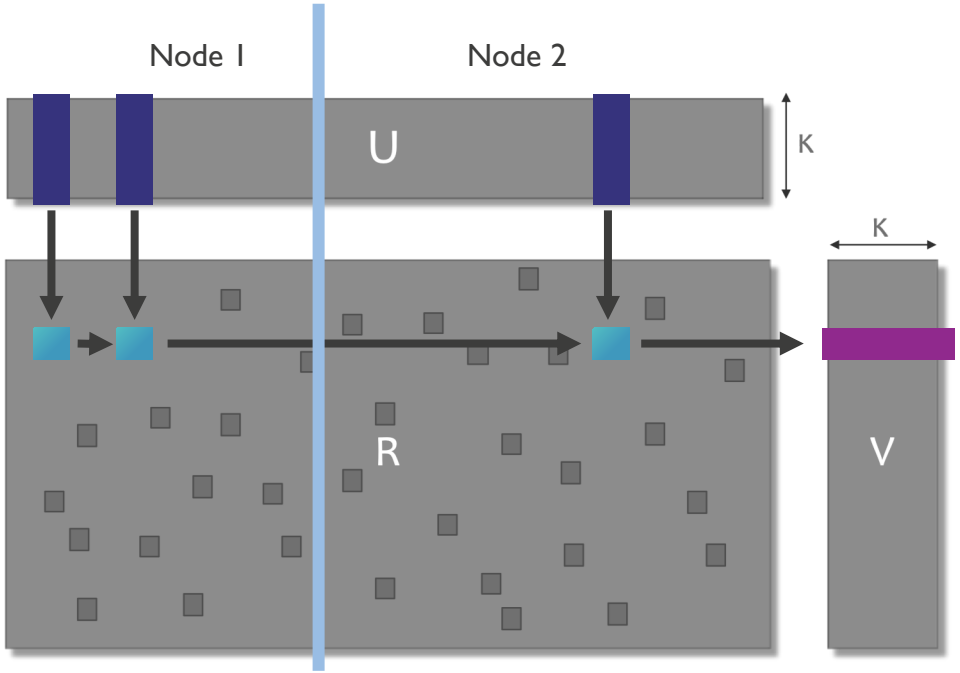
gaspi_notify

gaspi_write

ASYNCHRONOUS DISTRIBUTED BPMF

- Split both U/V, optimizing:
  1. Load balance (# rows, #nnz)
  2. Communication

- Basic Pattern GASPI
  - Compute a column of U/V
  - Send early

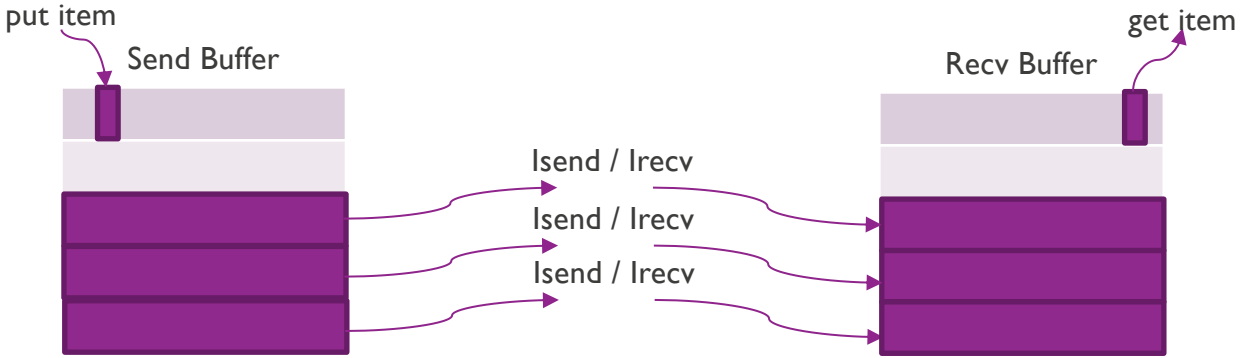


CHALLENGES USING MPI

- MPI is not thread-safe by default
  - Multiple work threads, one MPI thread
- MPI calls have high overhead
  - Buffer before send
- Many possible MPI primitives
  - MPI_Bcast
    - simple, synchronous, does not scale well
  - MPI_Put
    - need to split U in multiple MPI Windows, one window per peer
    - actual MPI work delayed until end of epoch
  - MPI_Isend/Irecv
    - best at doing background work
    - many irecvs/isends in flight

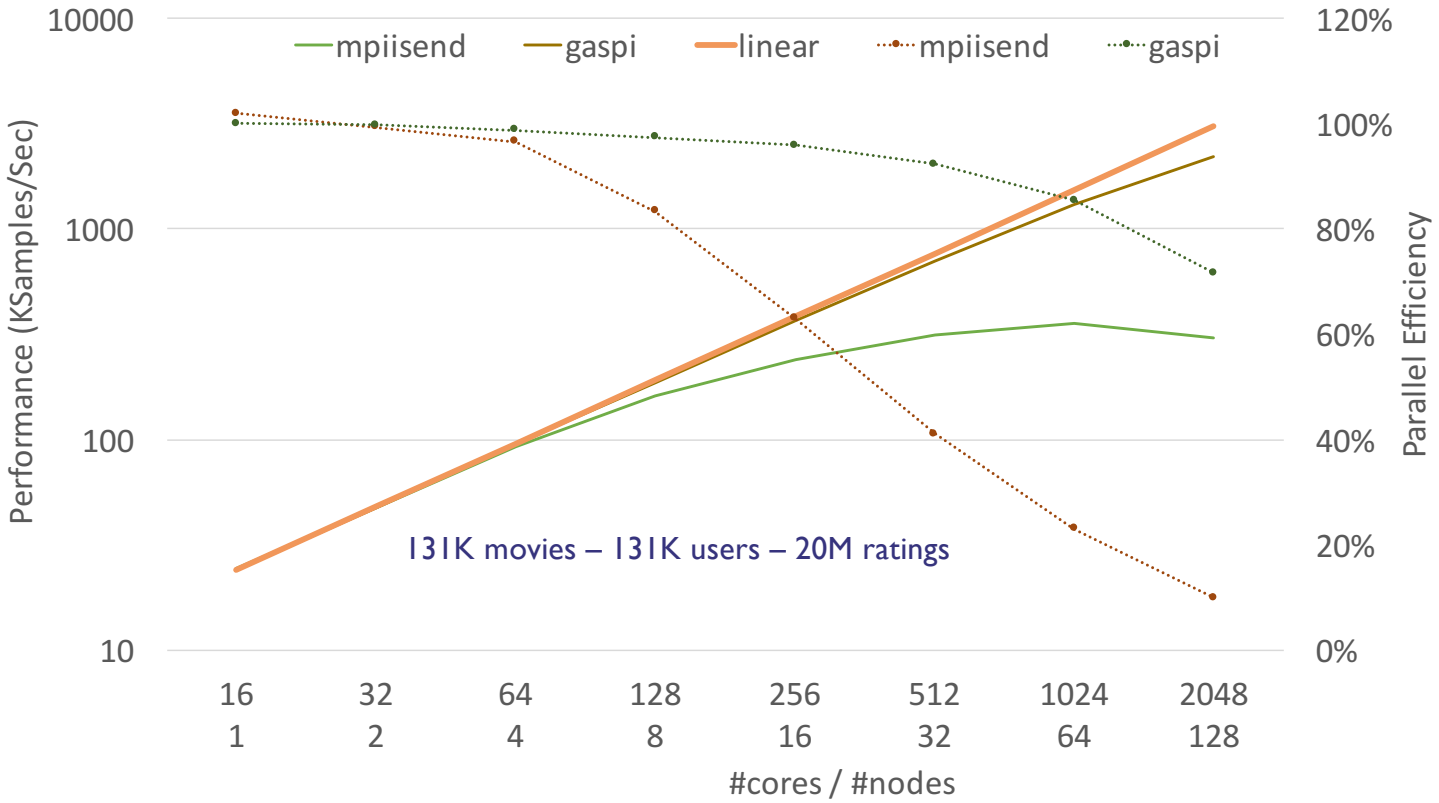
CURRENT BEST MPI IMPLEMENTATION

- Buffered ISend/IRecv
  - One buffer-pair per send-receive pair
  - Several chunks per buffer
  - Several items per chunk

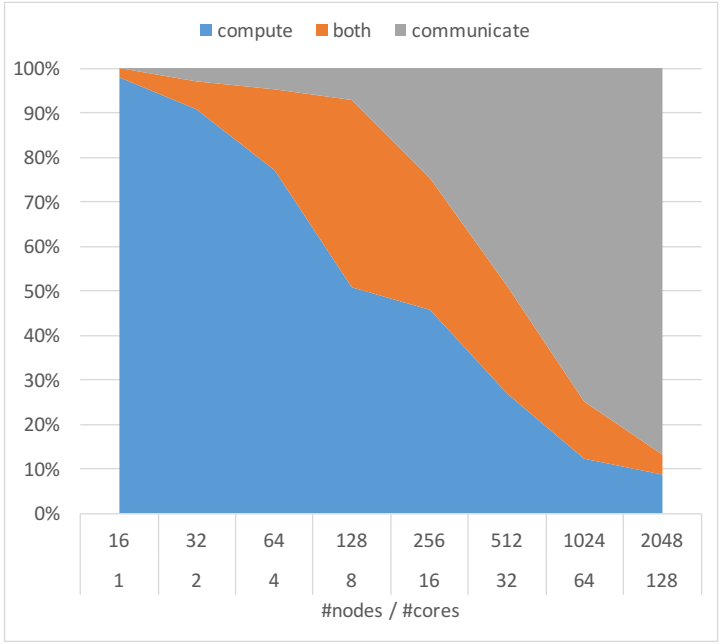


- 5 chunks → maximally 5 Isend/Irecv in flight
- 100 items per chunk

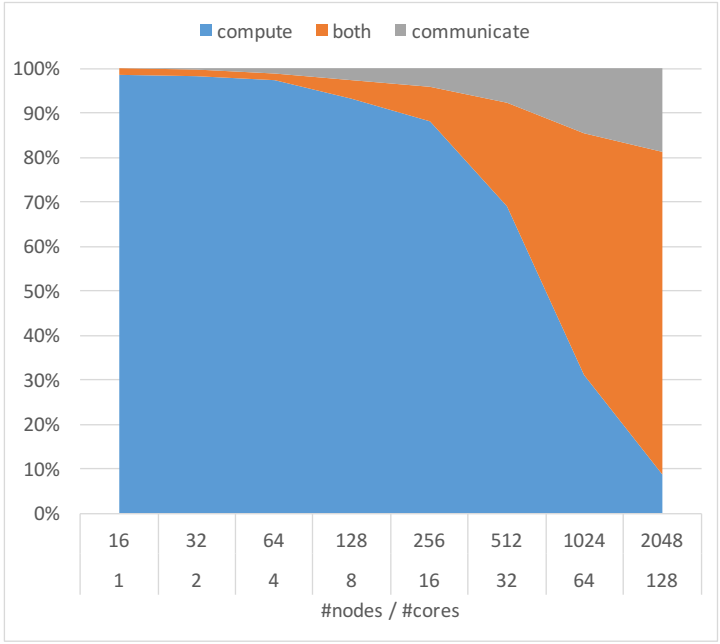
DISTRIBUTED PERFORMANCE – 128 NODES



COMM / COMP OVERLAP



MPI



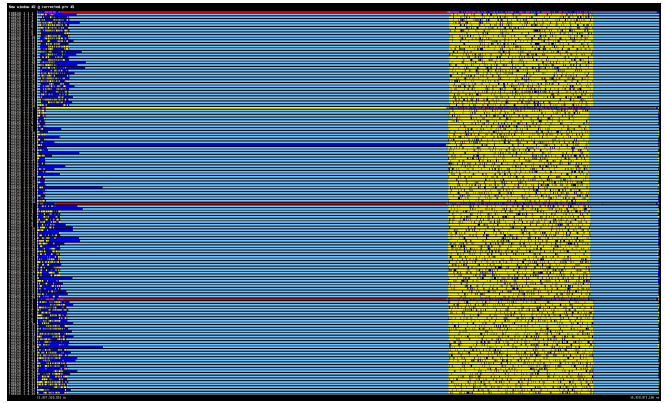
GASPI

POP COE

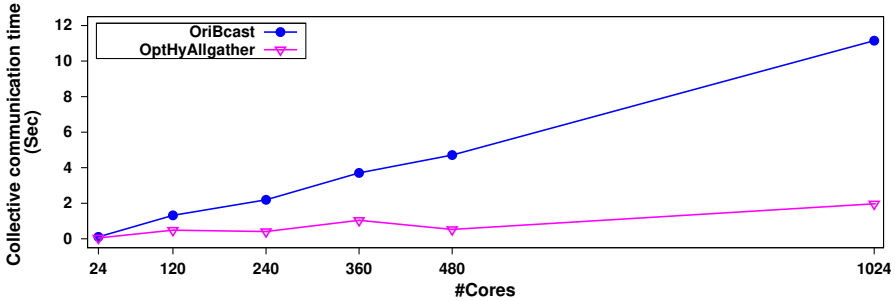
- HPC Expertise in ExCAPE (Low to High)
  - Pharma partners
  - Machine Learning Partners
  - HPC Partners (e.g IMEC)
  - POP Partners

- We learned
  - Tools: Extrae, Paraver
  - OpenMP tasks
  - MPI Asynchronous collectives

POP COE HELP

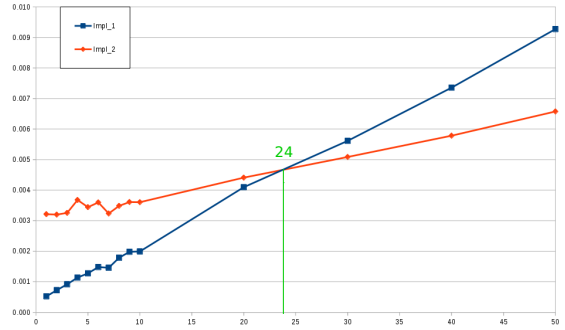


1. Audit Report
   - Load imbalanced detected using Extrae and Paraver
2. OpenMP Optimizations
   - Load imbalance fixed with extra arallelism using OpenMP tasks
3. Asynchronous MPI collectives



H L R S

Optimization: best values of breakpoint 1

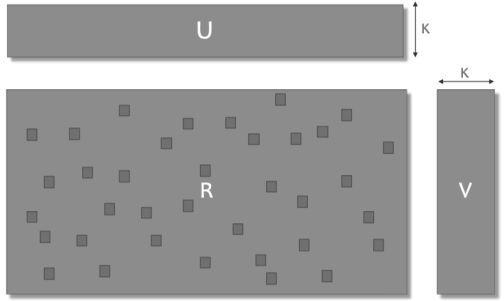


imec

CONCLUSIONS OPTIMIZING BPMF

- Reduced runtime on industrial data

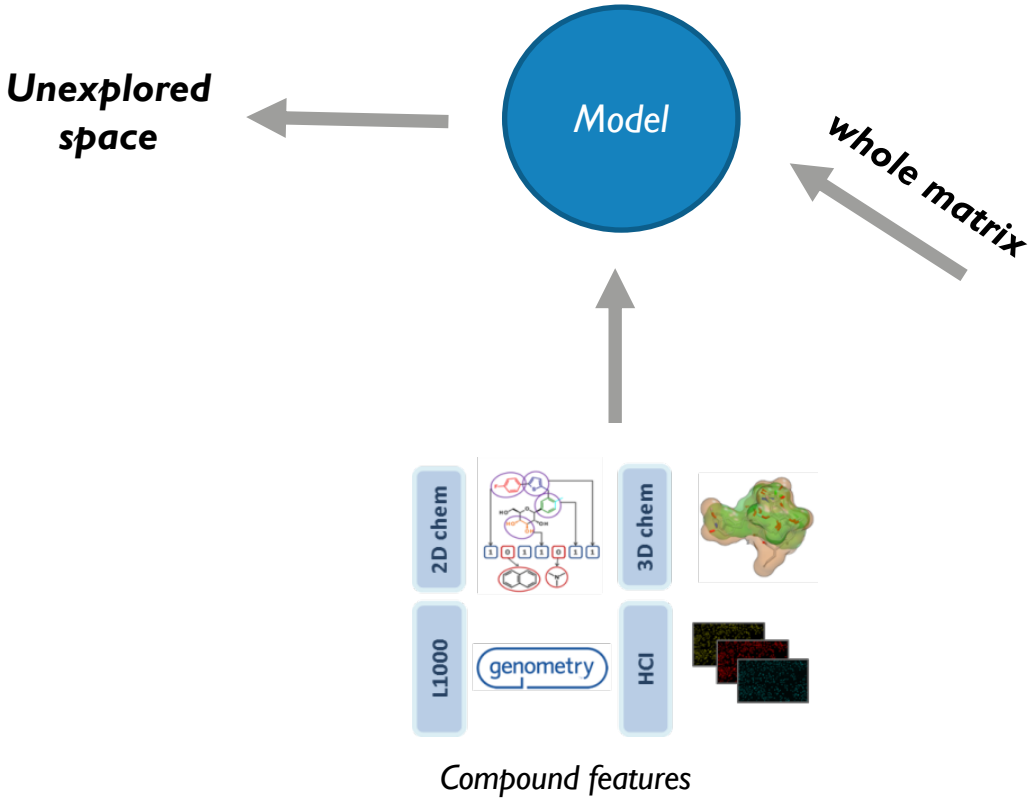
Parallelism	Time I Run
Single node - Julia	15 days
Single node - C++ & TBB	1.5 hours
Distributed - C++ & TBB & GASPI	5 minutes

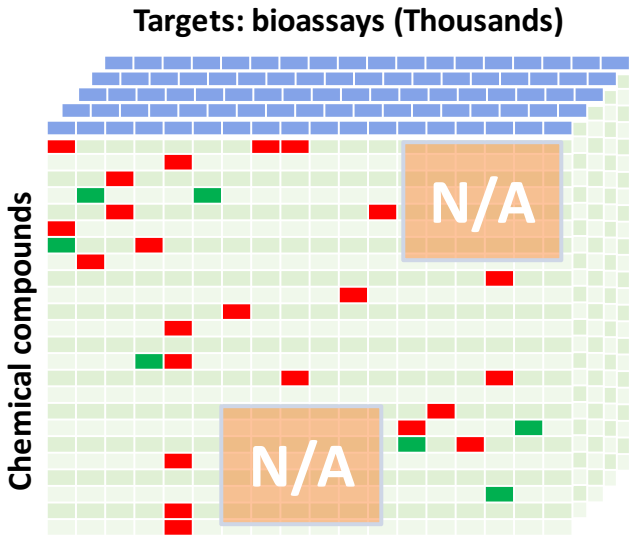


- Parallel efficiency is important
  - Load Balancing and Communication Hiding

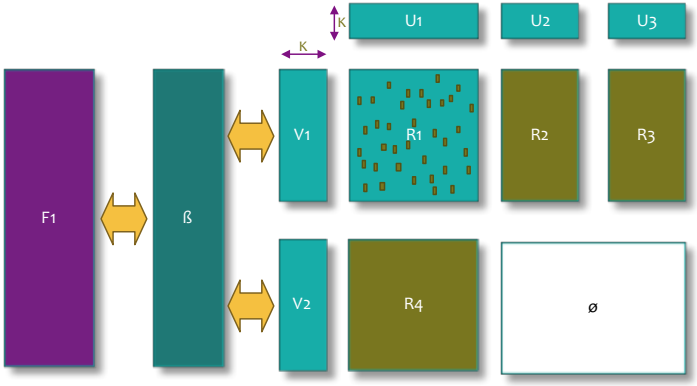
- BPMF Released on GitHub
  - https://github.com/ExaScience/bpmf

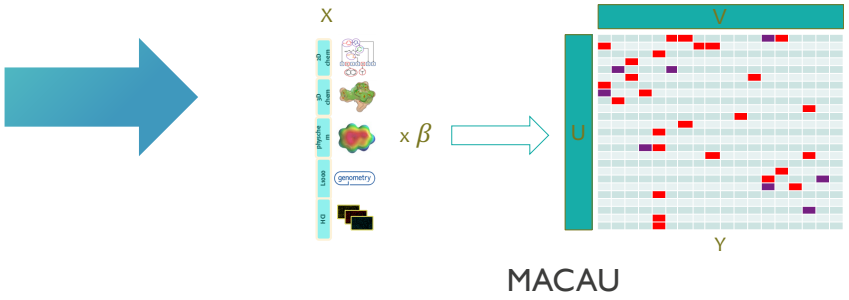
MULTI TARGET CHEMOGENOMICS



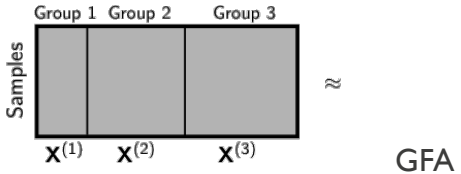


EXCAPE SMURFF FRAMEWORK

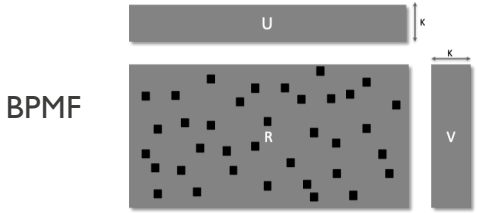




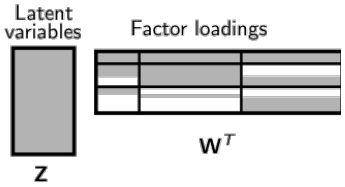
MACAU



GFA



BPMF



FROM BPMF TO SMURFF

SINGLE NODE, USER-FRIENDLY VERSION

- Versatile: many matrix & tensor factorization methods
- Maintainable: C++ with OpenMP and MPI
- Easy to use: `conda install smurff`
- Open source: https://github.com/ExaScience/smurff

A first example running SMURFF

In this notebook we will run the BPMF algorithm using SMURFF, on compound-activity data.

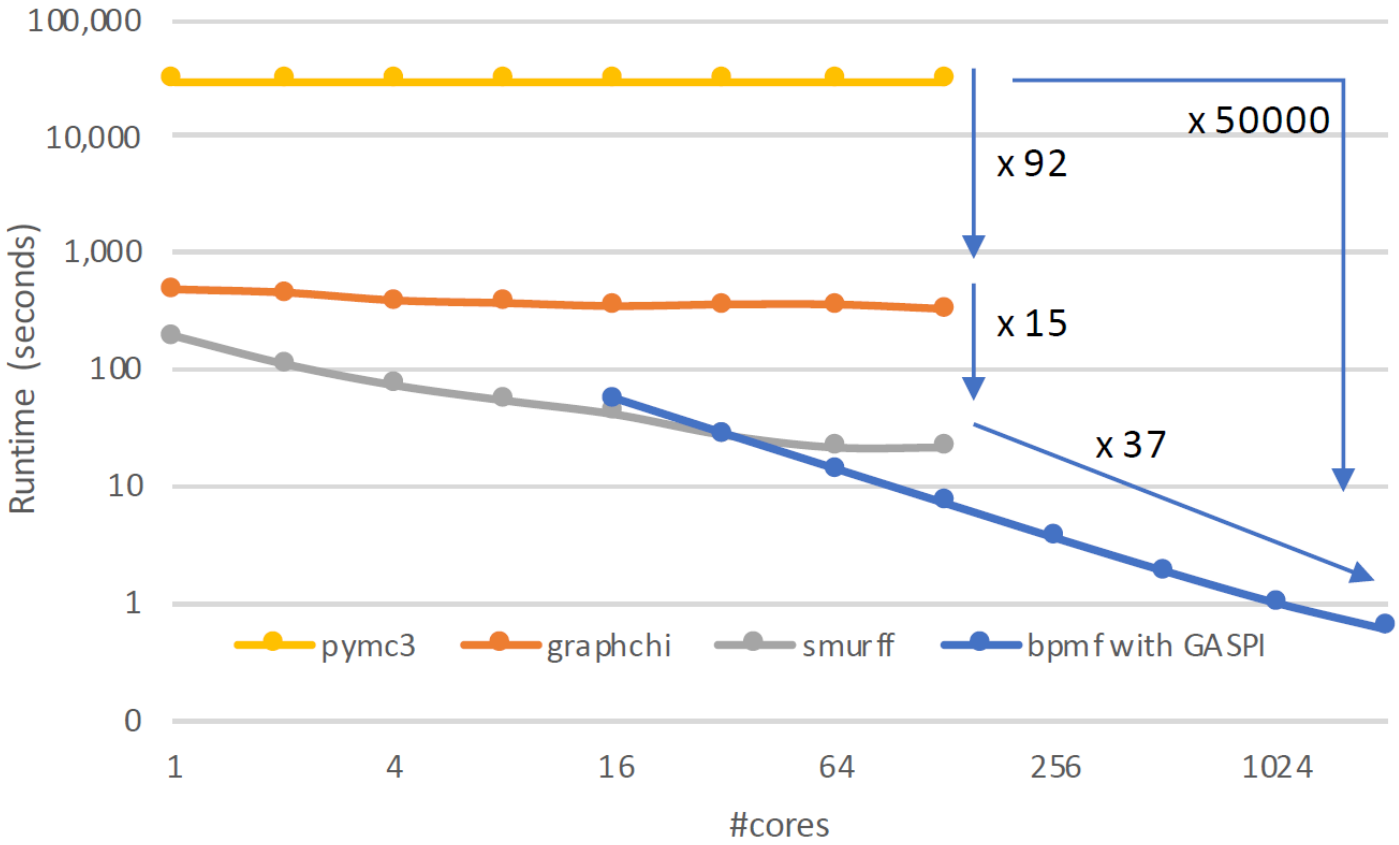
Downloading the data files

In these examples we use ChEMBL dataset for compound-proteins activities (IC50). The IC50 v
downloaded using this smurff function:

```
l]: import smurff

    ic50_train, ic50_test, ecfp = smurff.load_chembl()
```

SMURFF PERFORMANCE IS STILL GOOD



ASYNCHRONOUS DISTRIBUTED BPMF WITH POSTERIOR PROPAGATION

Latent factors
Global: $\mathbf{X}_{N\times K}$
Local: $\widetilde{\mathbf{X}}^{ij}_{N_i\times K}$

$$\begin{bmatrix} \mathbf{W}^1 & \mathbf{W}^2 & \mathbf{W}^3 & \mathbf{W}^4 \\ \widetilde{\mathbf{W}}^{i1} & \widetilde{\mathbf{W}}^{i2} & \widetilde{\mathbf{W}}^{i3} & \widetilde{\mathbf{W}}^{i4} \end{bmatrix}$$

Loading matrix
Global: $\mathbf{W} \in \mathbf{R}^{D\times K}$
Local: $\widetilde{\mathbf{W}}^{ij} \in \mathbf{R}^{D_j\times K}$

$$\begin{bmatrix} \mathbf{X}^1 & \widetilde{\mathbf{X}}^{1j} \\ \mathbf{X}^2 & \widetilde{\mathbf{X}}^{2j} \\ \mathbf{X}^3 & \widetilde{\mathbf{X}}^{3j} \\ \mathbf{X}^4 & \widetilde{\mathbf{X}}^{4j} \end{bmatrix}$$

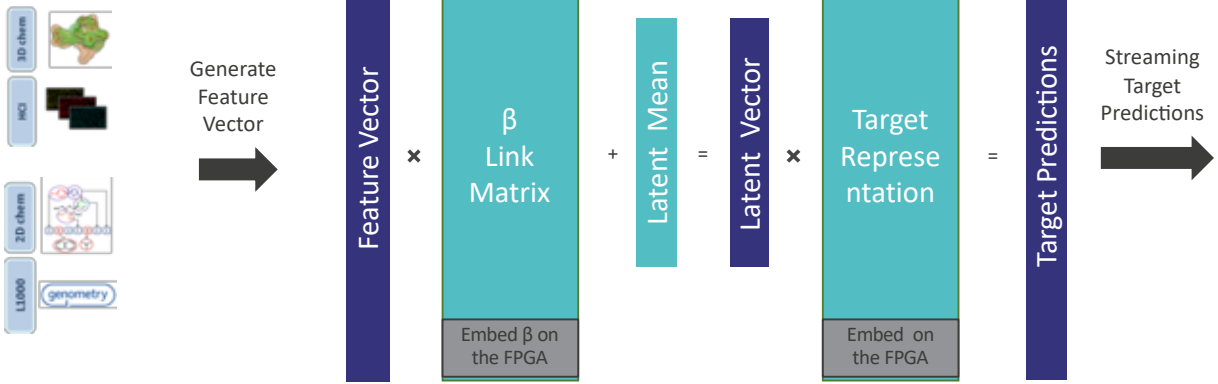
$$\begin{bmatrix} \mathbf{Y}^{11} & \mathbf{Y}^{12} & \mathbf{Y}^{13} & \mathbf{Y}^{14} \\ \mathbf{Y}^{21} & \mathbf{Y}^{22} & \mathbf{Y}^{23} & \mathbf{Y}^{24} \\ \mathbf{Y}^{31} & \mathbf{Y}^{32} & \mathbf{Y}^{33} & \mathbf{Y}^{34} \\ \mathbf{Y}^{41} & \mathbf{Y}^{42} & \mathbf{Y}^{43} & \mathbf{Y}^{44} \end{bmatrix}$$

Full data: $\mathbf{Y}_{N\times D}$
Subdata: $\mathbf{Y}^{ij}_{N_i\times D_j}$

The formulation of submodels in the third stage takes the same form as BPMF, but with different priors for model parameters, i.e.

$$\begin{aligned} \mathbf{y}_n^{(i,j)} &= \widetilde{\mathbf{W}}^{(i,j)}\widetilde{\mathbf{x}}_n^{(i,j)} + \epsilon_n,\ \epsilon_n \sim \mathcal{N}_{D_j}(0,\Sigma) \\ \mathbf{x}_n^{(i,j)} &\sim p(\mathbf{x}_n^{(i,j)}|\mathbf{X}^{(i,1)}) \sim \mathcal{N}_K(\mathbf{y}_n^{(i,j)}|\mu_{\mathbf{X}^{(i,1)}}, \Sigma_{\mathbf{X}^{(i,1)}}) \\ \mathbf{w}_d^{(i,j)} &\sim p(\mathbf{w}_d^{(i,j)}|\mathbf{W}^{(1,j)}) \sim \mathcal{N}_K(\mathbf{w}_d^{(i,j)}|\mu_{\mathbf{w}_d^{(1,j)}}, \Sigma_{\mathbf{w}_d^{(1,j)}}) \end{aligned} \quad (2)$$

VIRTUAL MOLECULE SCREENING
ON CPU, GPU, AND FPGA



EURO EXA
FUNDED BY THE EUROPEAN UNION

EPEEC
PRODUCTIVITY AT EXASCALE

CONCLUSIONS

- Compound Activity Prediction in the ExCAPE project
- Matrix Factorization

- BPMF with GASPI optimized by POP
- SMURFF - https://github.com/ExaScience/smurff

- Virtual Molecule Screening on GPU and FPGA