# Data Placement Optimization in GPU Memory Hierarchy Using Predictive Modeling

Larisa Stoltzfus[*], Murali Emani, Pei-Hung Lin, Chunhua Liao
[*]University of Edinburgh (UK), Lawrence Livermore National Laboratory

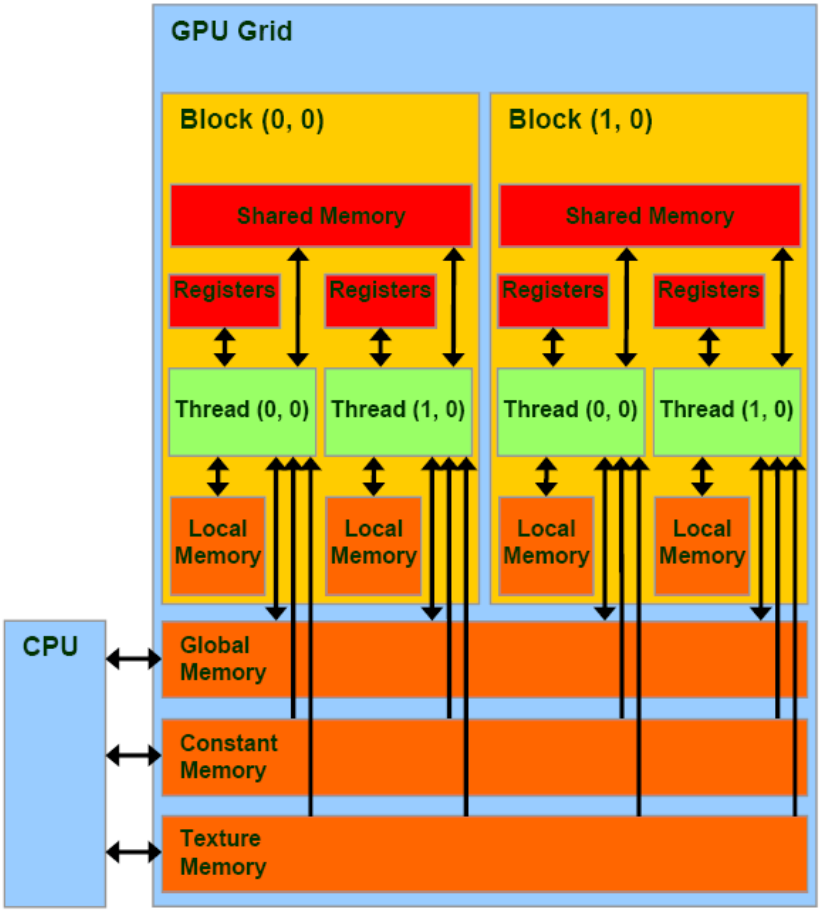
MCHPC'18: Workshop on Memory Centric High Performance Computing

LLNL-PRES-761162 This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

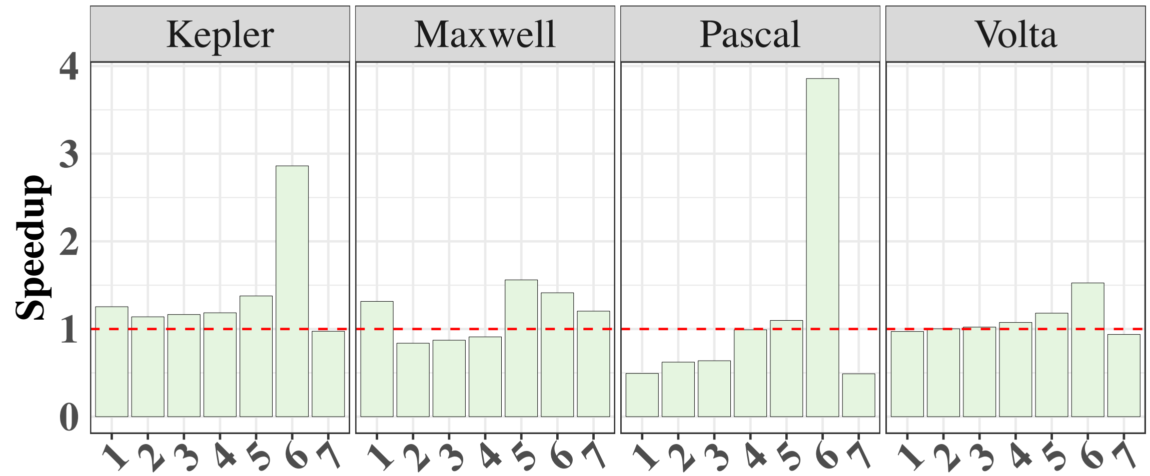
# Complex Memory Hierarchy on GPUs

- GPUs can greatly improve performance of HPC applications, but can be difficult to optimize for due to their complex memory hierarchy
- Memory hierarchies can change drastically from generation to generation
- Codes optimized for one platform may not retain optimal performance when ported to other platforms

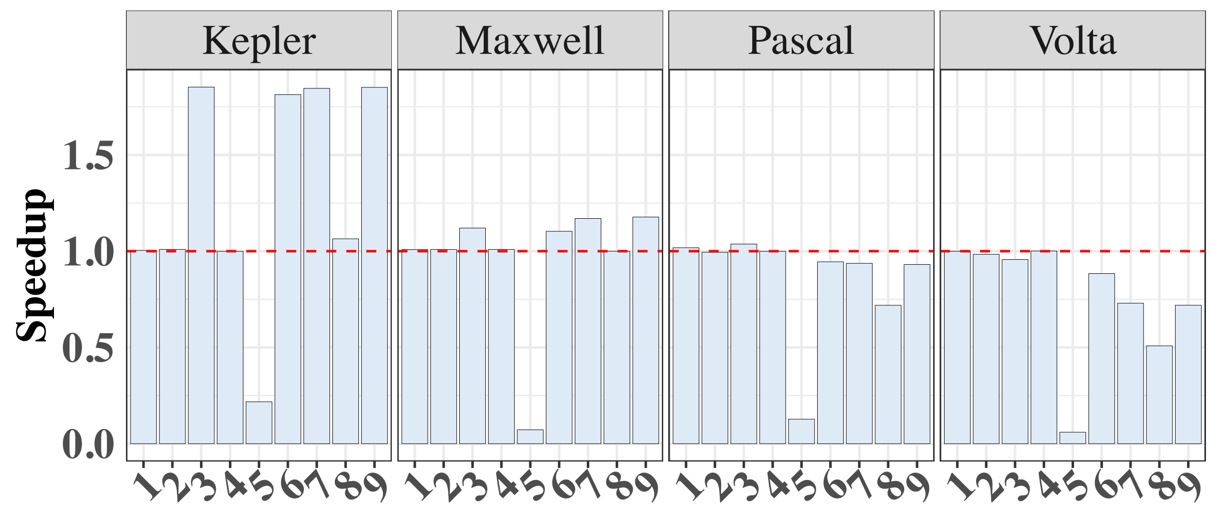


# Performance can vary widely depending on data placement as well as platform

Matrix-Matrix Multiplication




Sparse Matrix-Vector Multiplication



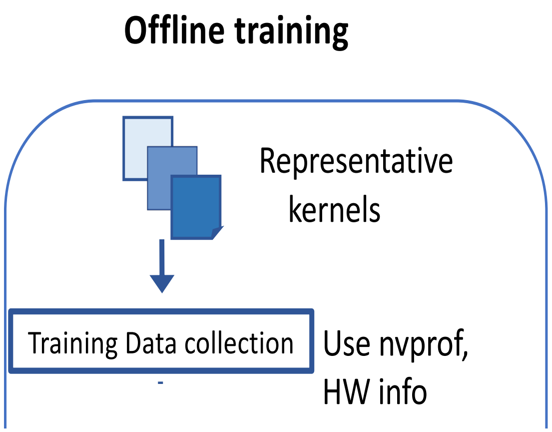
# Challenges

- Different memory variants (global/ constant/ texture/ shared) can have significant impact on program performance
- But identifying the best performing variant is non-obvious and complex decision to make
- Given a default global variant, can the best performing memory variant be automatically determined?

# Proposed Solution

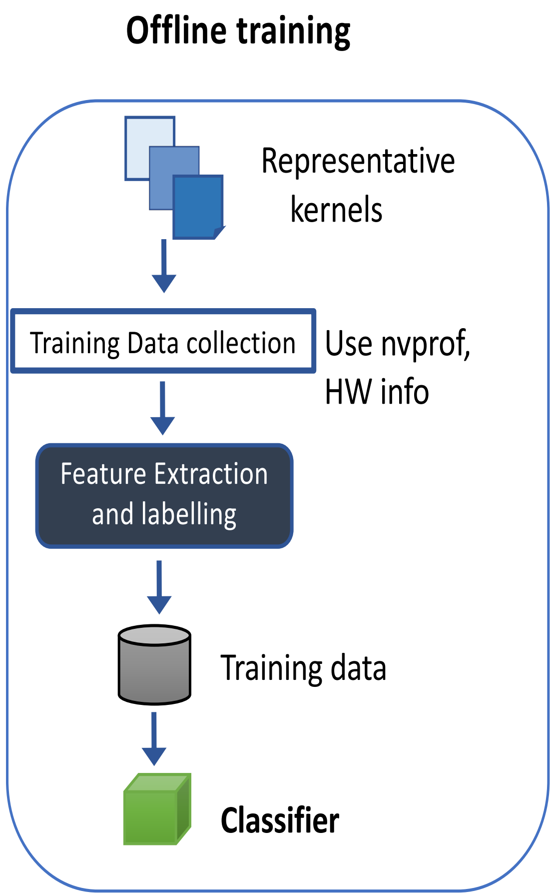
- Use machine learning to develop a predictive model to determine the best data placement for a given application on a particular platform
- Use the model to predict best placement
- Involves three stages:
  - offline training
  - feature and model selection
  - online inference

# Approach




**Offline Training** - Data collection of nvprof metrics and events

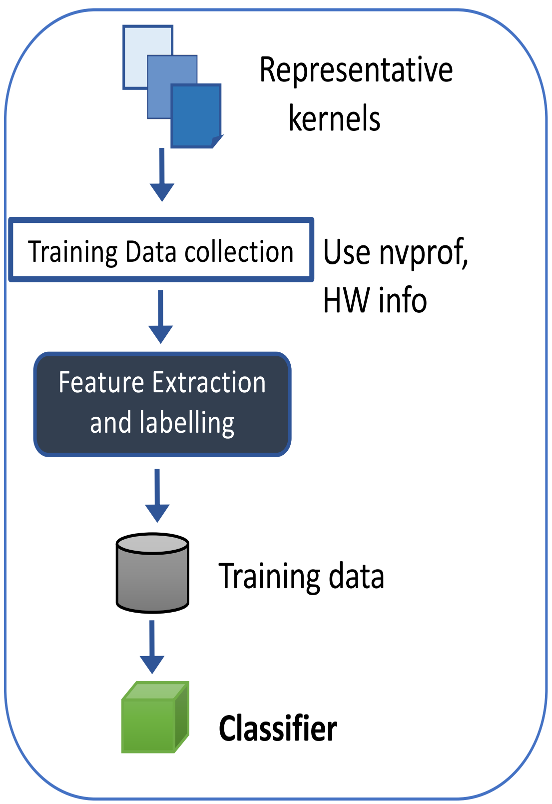
# Approach



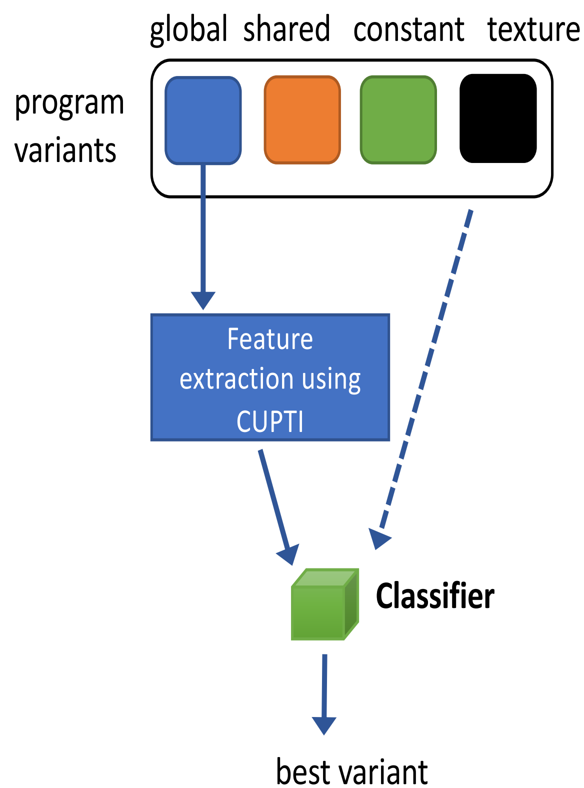

**Model Building** - Determine best version, features and model

# Approach

**Offline training**




**Online inference**




**Online Inference:** Use model to determine best placement in run-time

# Methodology

In order to build the model:

- 4 different generations of NVIDIA GPUs were used:
  - Kepler
  - Pascal
  - Maxwell
  - Volta

- 8 programs **X** 3 input data sizes **X** 3 thread block sizes **X** 4 variants

MD, SPMV, CFD, MM, ConvolutionSeparable, ParticleFilter etc.

# Offline Training

- Metric and event data from nvprof from global variant along with hardware data were collected
- Best performing variant (class label) for each version run was appended
- Benchmarks were run 10 times on each platform, with 5 initial iterations to warm up the GPU

# Feature Selection

- Number of features narrowed down to 16 from 241 using correlation-based feature selection algorithm (CFS).
- A partial list:

| Feature Name | Meaning |
|---|---|
| achieved_occupancy | ratio of average active warps to maximum number of warps |
| l2_read_transactions, l2_write_transactions | Memory read/write transactions at L2 cache |
| gld_throughput | global memory load throughput |
| warp_execution_efficiency | ratio of average active threads to the maximum number of threads |

# Model Selection

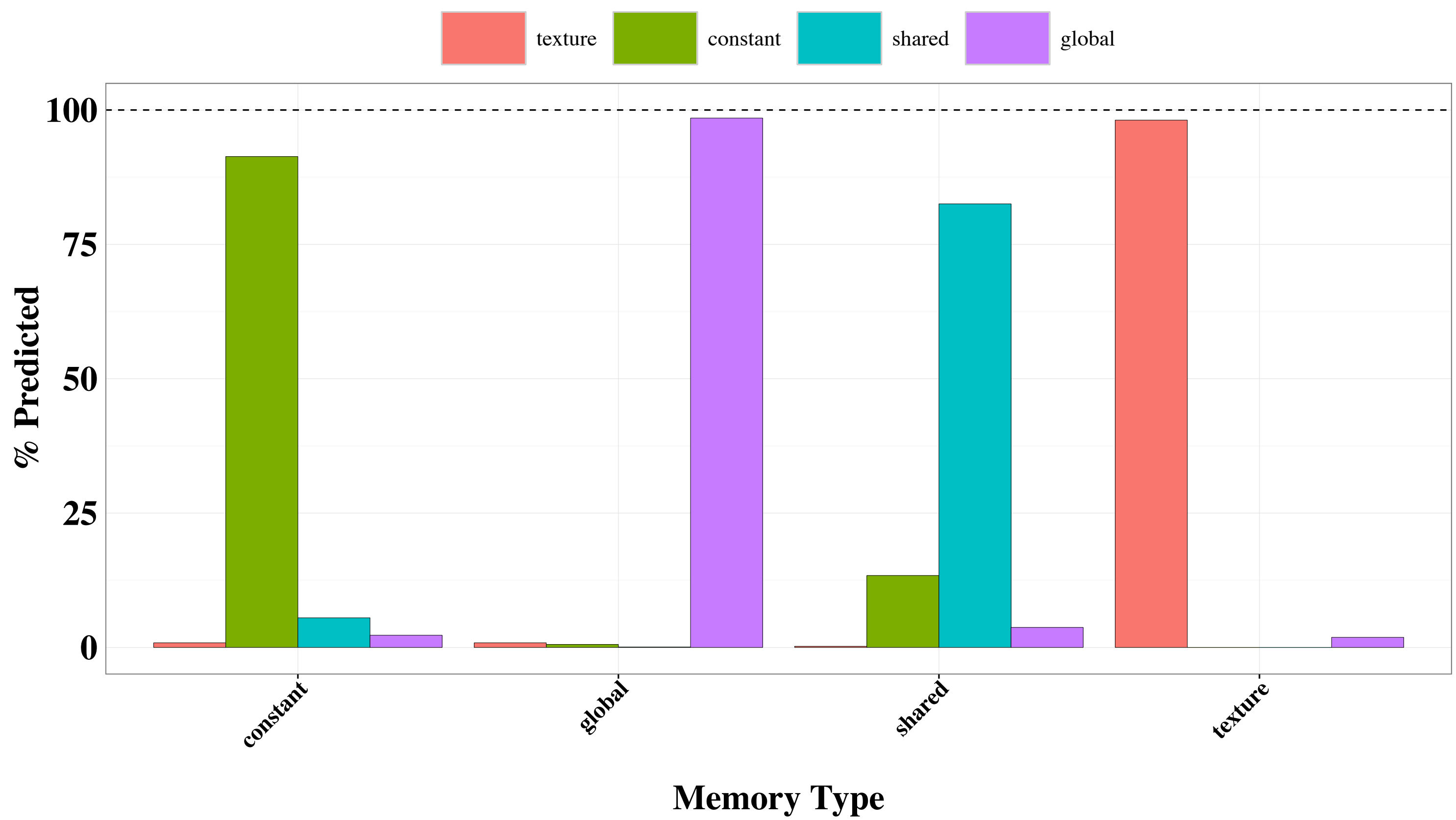
- Used 10-fold cross validation during evaluation
- Overall, decision tree classifiers showed great promise (>95% accuracy in prediction)

| Classifier | Prediction Accuracy (%) |
|---|---|
| RandomForest | 95.7 |
| LogitBoost | 95.5 |
| IterativeClassifierOptimizer | 95.5 |
| SimpleLogistic | 95.4 |
| JRip | 95.0 |

# Runtime Prediction

- The classifier JRIP was selected from the group of top five performing classifier models
- JRIP is a propositional rule learner, which results in a decision tree
- The model then reads in input from CUPTI calls - the API for nvprof - which can access hardware counters in real-time and outputs its class

# Preliminary Results




- Results from this initial exploration show that there is great potential for predictive modeling for data placement on GPUs
- Overall 95% accuracy achievable, but this is higher for global and texture memory best performers

# Runtime Validation

- The JRIP model was tested out on a new benchmark - an acoustic application
- The model was successfully able to correctly predict the best performing version on two platforms

# Limitations

- Currently, all versions need to be pre-compiled for run-time prediction, ideally it would be better to have model built into a compiler
- CUPTI calls are slow and require as many iterations as metrics and events to collect
- This would acceptable for benchmarks with many iterations, but for other kinds a workaround would need to be made

# Conclusion

- Machine learning has shown great potential for data placement prediction on a range of applications
- More work needs to be done to acquire hardware counters from applications in a timely manner
- Approach could be reused for other optimizations such as data layouts.

Lawrence Livermore
National Laboratory

| Version | Description |
|---|---|
| 1 | rows array in shared |
| 2 | rows array in constant |
| 3 | vector array in texture1D, rows in shared |
| 4 | matrix values in texture1D |
| 5 | vector array in constant, rows in texture1D |
| 6 | vector array in texture |
| 7 | matrix values and columns in texture1D, rows in constant |
| 8 | matrix values, columns and vector in texture1D |
| 9 | matrix values, columns, rows and vector in texture1D |

**Table 2: Memory configuration details of SPMV benchmark**