



Sandia
National
Laboratories

Automatic Discovery of Implementation Rules for Fast GPU + MPI Operations

Carl Pearson, Karen Devine, Aurya Javeed

Sandia National Labs

SIAM Parallel Processing 2022

MS61 Experiences in Developing GPU Support for DOE Math Libraries

This work is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231 using NERSC award ERCAP0019623.



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Automatic Discovery of Implementation Rules for Fast GPU + MPI Operations



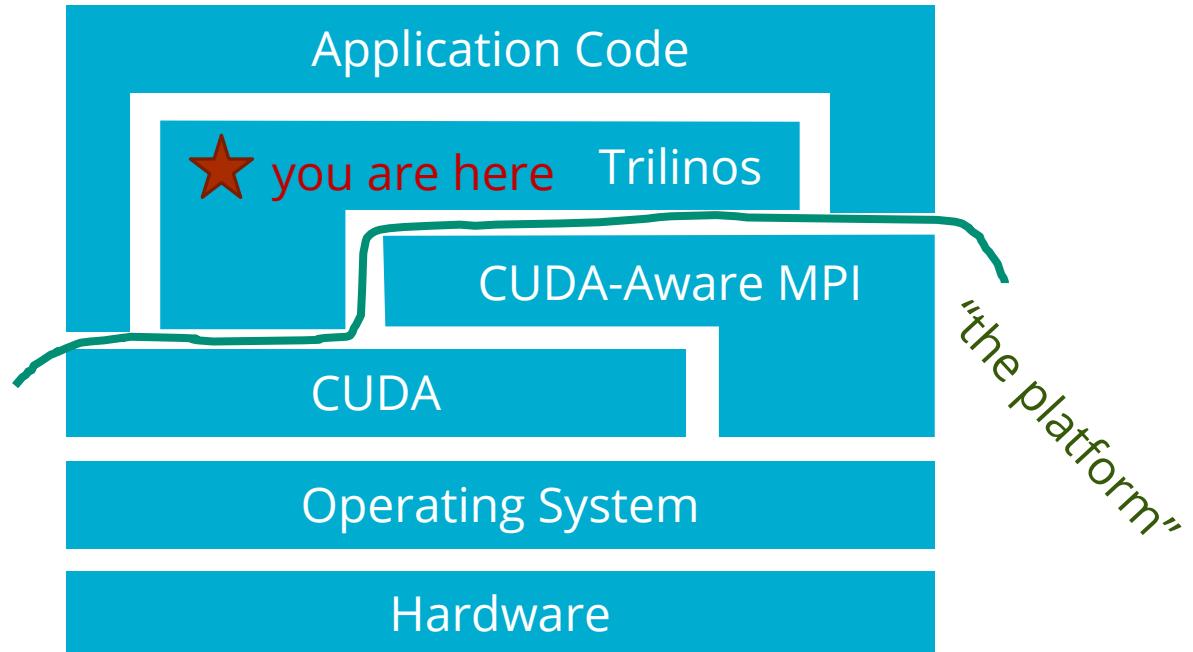
- Fast libraries for heterogeneous architectures
 - Mapping computation onto processors
 - Choosing communication strategy
 - Unpredictable performance interaction
- Prototype automatic tooling for discovering important design decisions
 - Reduced developer effort for performance on new systems
 - Maintain human provenance of library design
 - e.g. Modernize Tpetra MPI+GPU distributed linear algebra operations

Key Challenge	How it's Done
Large Design Space	<ul style="list-style-type: none">• Express operation as a directed acyclic graph (DAG) of operations• Monte-Carlo Tree Search (MCTS) to identify and explore regions of interest
Extract performance insight	<ul style="list-style-type: none">• Empirical benchmarking• Feature vector for each implementation• Decision tree training for design rules

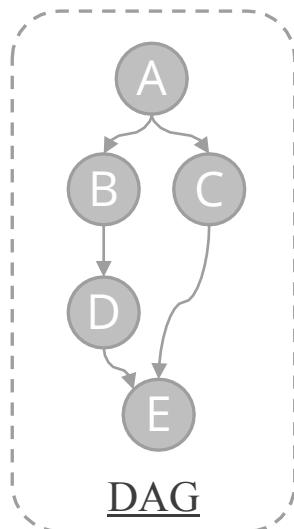
Initial results pass “sniff test,” working on broader experiments and quantitative evaluation

Libraries are built on existing lower-level primitives

- Our libraries (and applications) are combinations of existing library and vendor operations
 - and code to coordinate them
 - and code to implement custom behavior
- Performance changes at many layers for new platforms
 - new hardware,
 - new CUDA version,
 - new OS version,
 - etc.



Prototype Implementation in C++ and Python

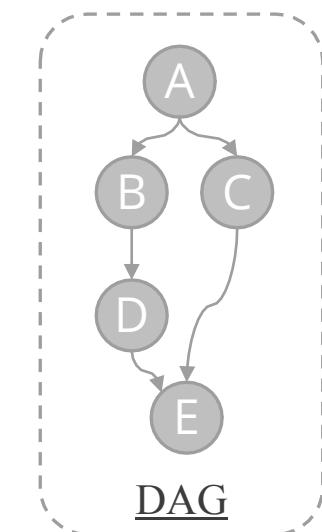


DAG
DAG of
operations
describes design
space

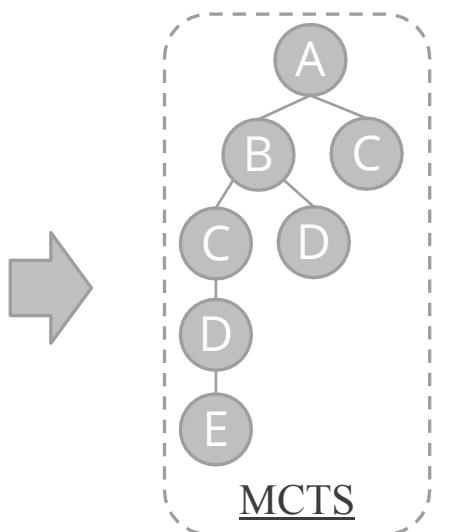
C++ / CUDA / MPI

Python / scikit-learn

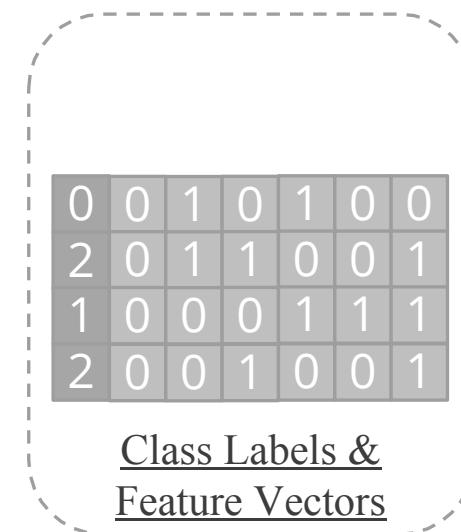
Prototype Implementation in C++ and Python



DAG of operations describes design space



MCTS searches order of operations and resource assignment

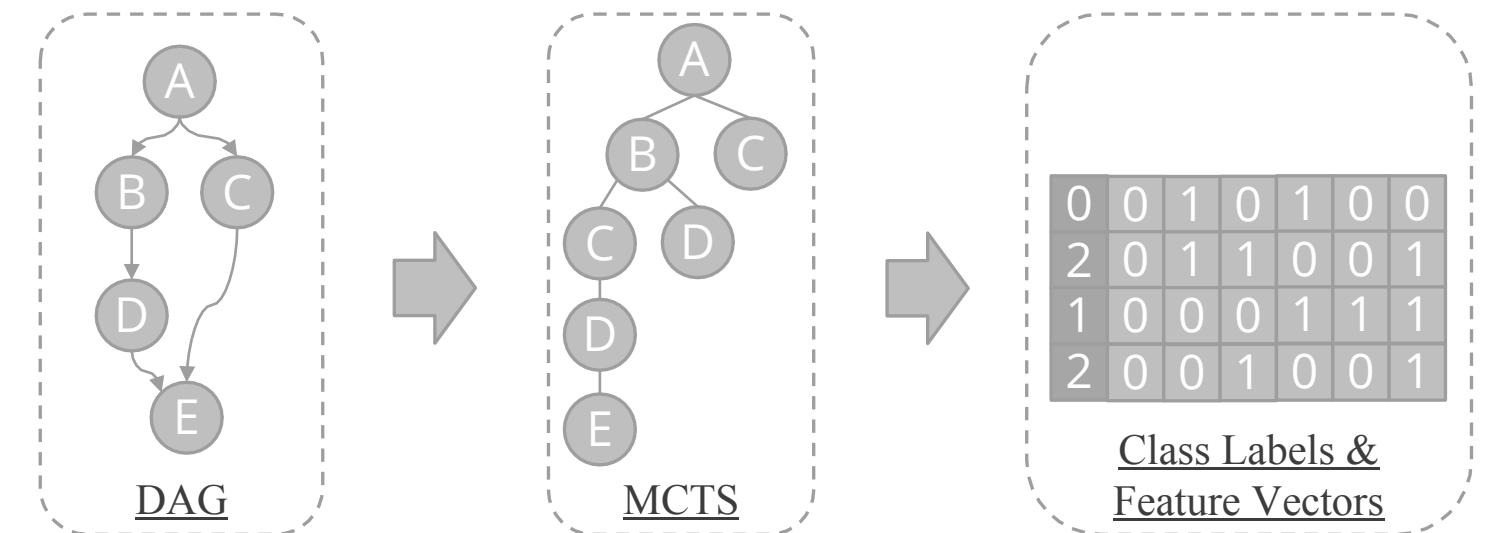


Sequence-to-vector transformation for labels

C++ / CUDA / MPI

Python / scikit-learn

Prototype Implementation in C++ and Python



DAG of operations describes design space

MCTS searches order of operations and resource assignment

Sequence-to-vector transformation for labels

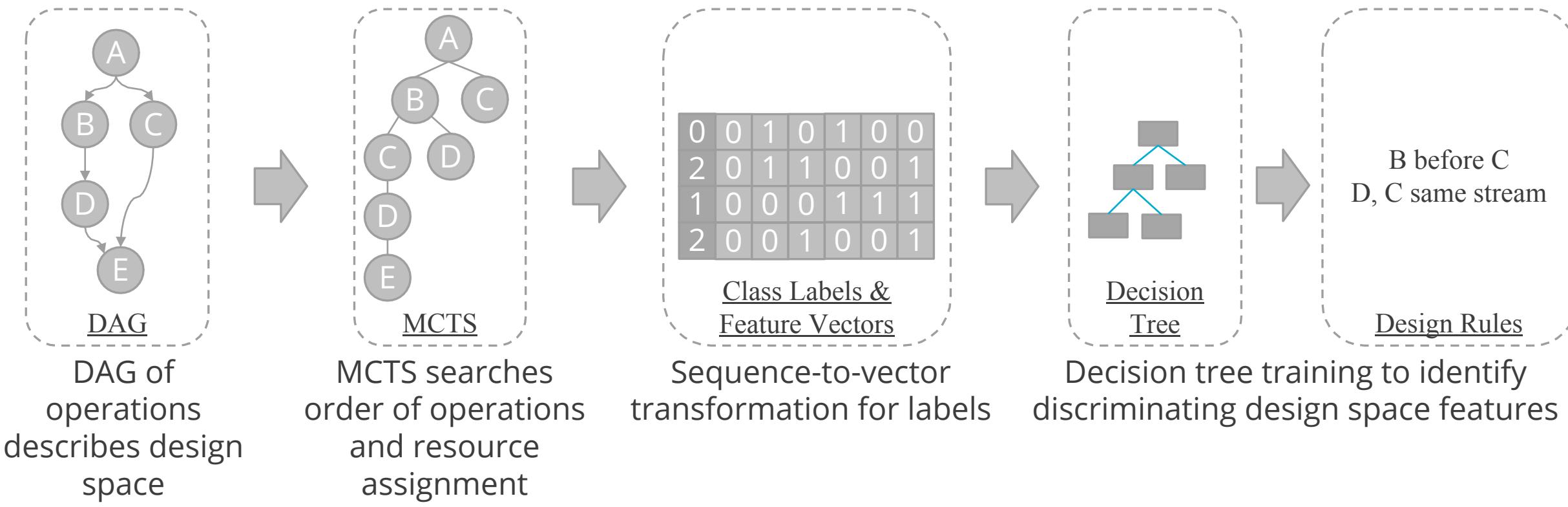
0	0	1	0	1	0	0
2	0	1	1	0	0	1
1	0	0	0	1	1	1
2	0	0	1	0	0	1

Class Labels & Feature Vectors

C++ / CUDA / MPI

Python / scikit-learn

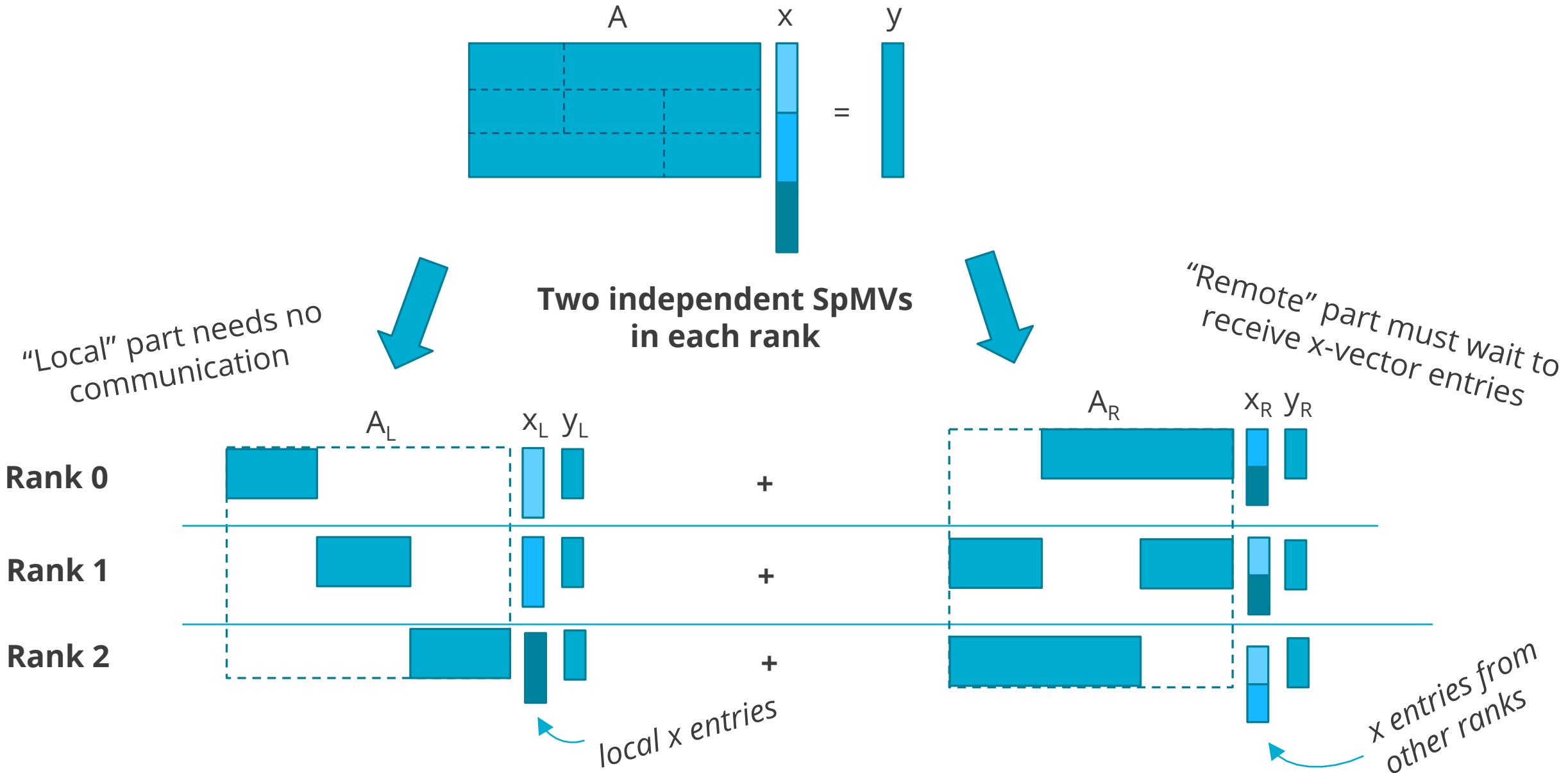
Prototype Implementation in C++ and Python



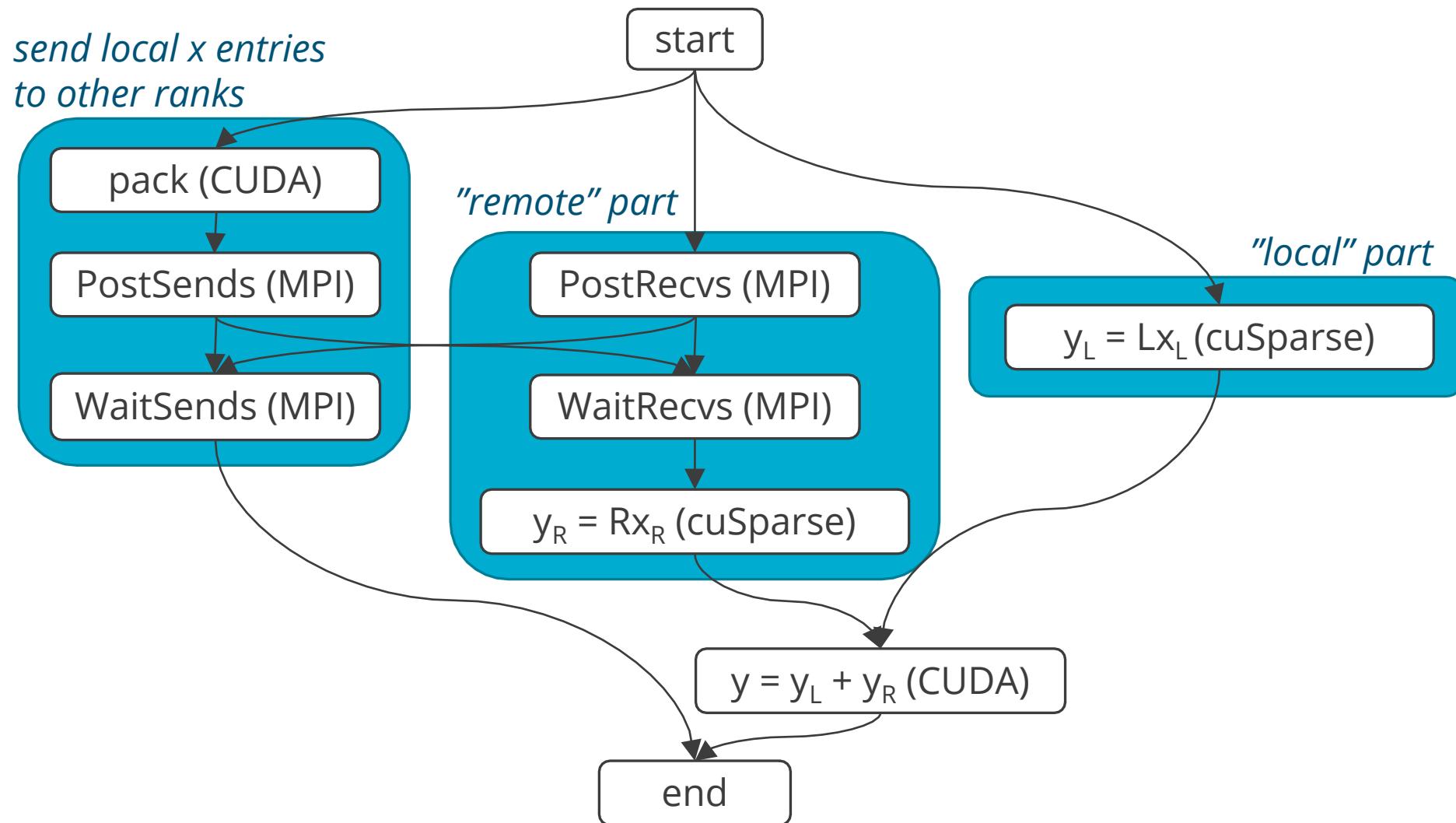
C++ / CUDA / MPI

Python / scikit-learn

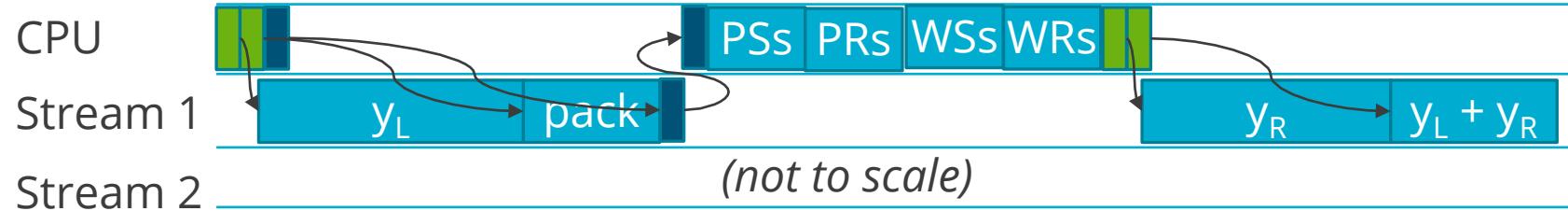
Example: Distributed SpMV



DAG represents primitive operations and their dependences

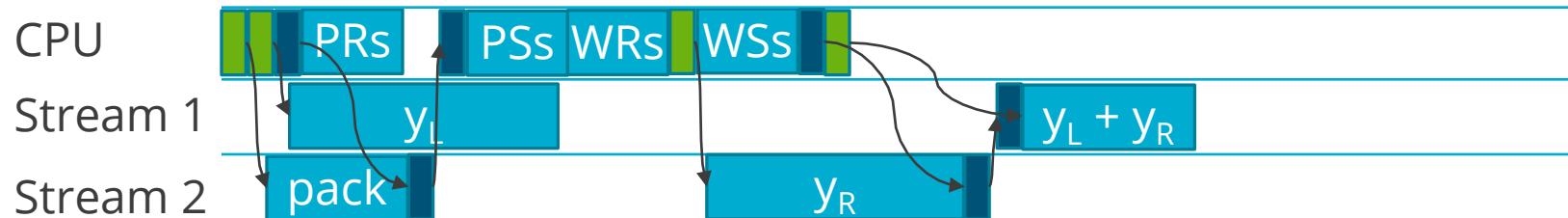


Design Space: Order of Operations, Resource Assignment, and Synchronization



- Different resource assignments require different synchronization
- May improve GPU utilization or communication/computation overlap, but increases required operations

█ kernel launch
█ sync ops
█ application operations



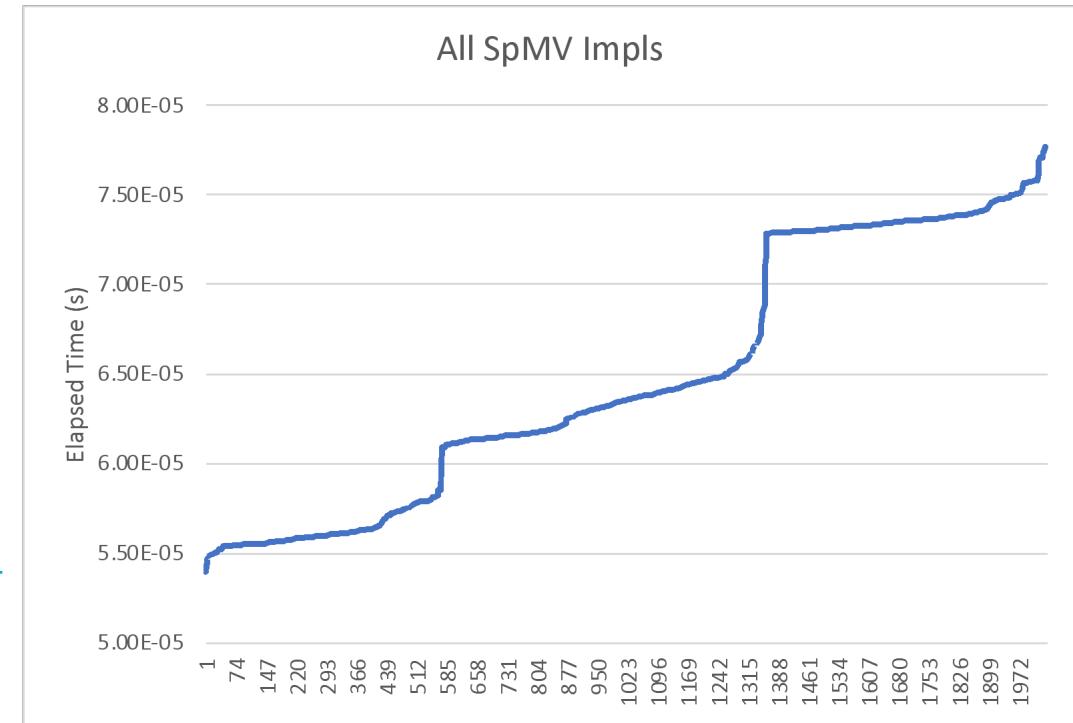
Need to Discover Important Design Decisions



- Some choices matter a lot
- Many choices do not matter at all
- input- and system-dependent
- Large design space: lots of expert time to evaluate and implement for each target platform
- Monte-Carlo Tree Search to focus on valuable decisions

{order of operations} \times
 {stream assignments} \times
 {synchronizations}

1.45x speedup

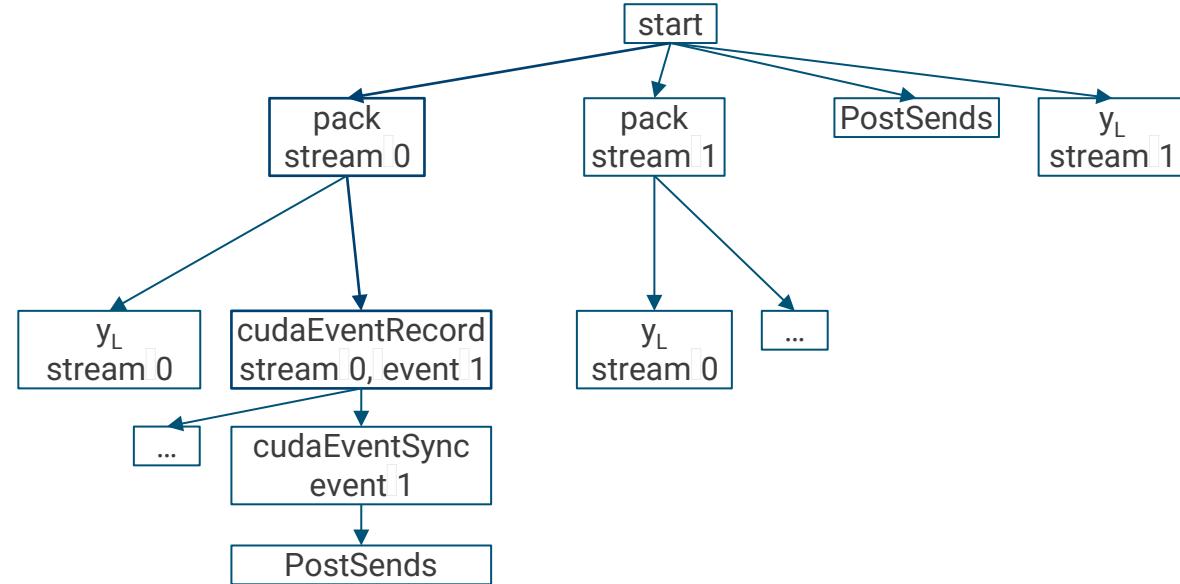


2036 implementations

MCTS Represents Search State in a Tree



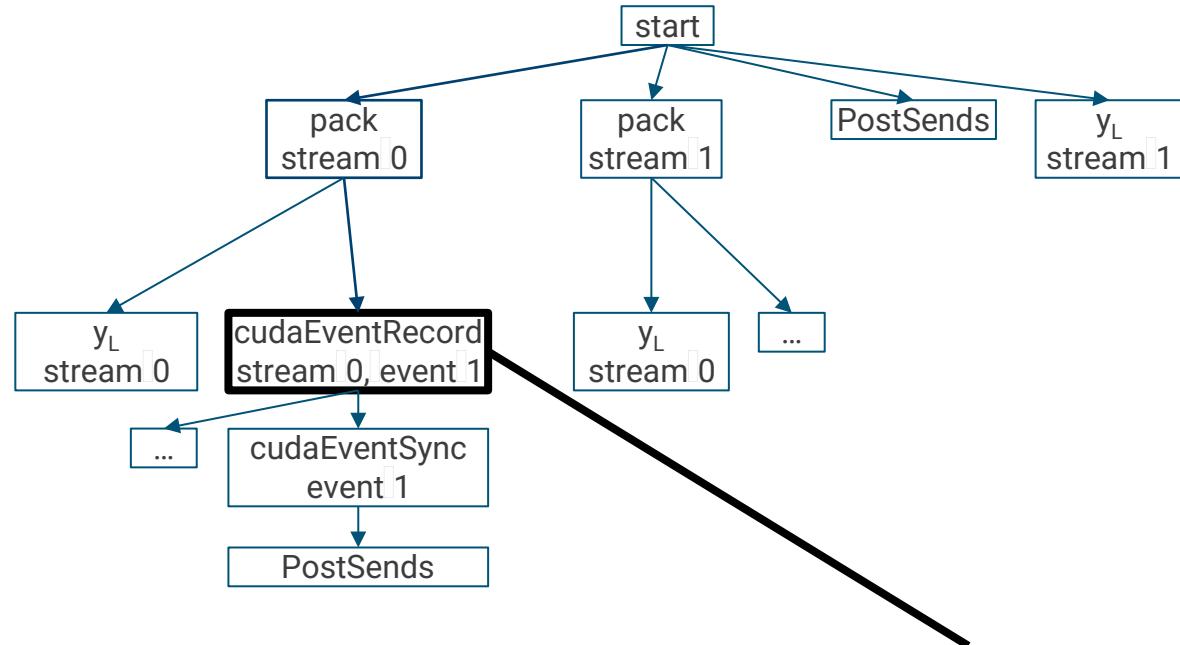
State space search is stored in a tree



MCTS Represents Search State in a Tree



State space search is stored in a tree



Each node is an operation and resource assignment

From DAG, or synchronization operation

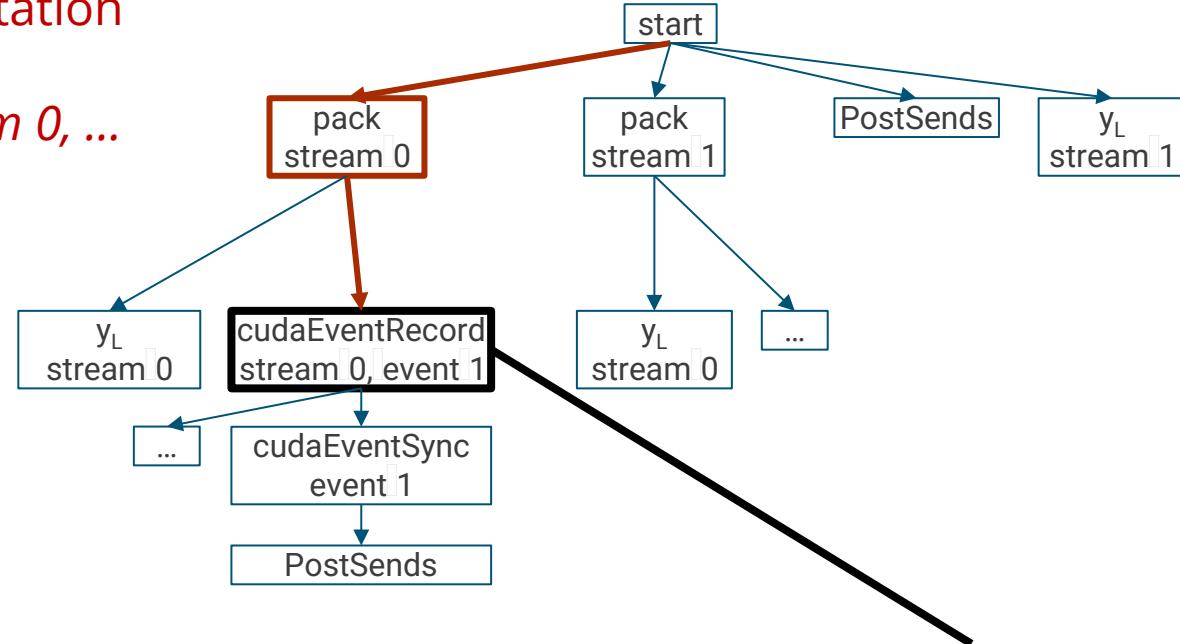
MCTS Represents Search State in a Tree



State space search is stored in a tree

Path is the beginning of an implementation

pack in stream 0, record event 1 in stream 0, ...



Each node is an operation and resource assignment

From DAG, or synchronization operation

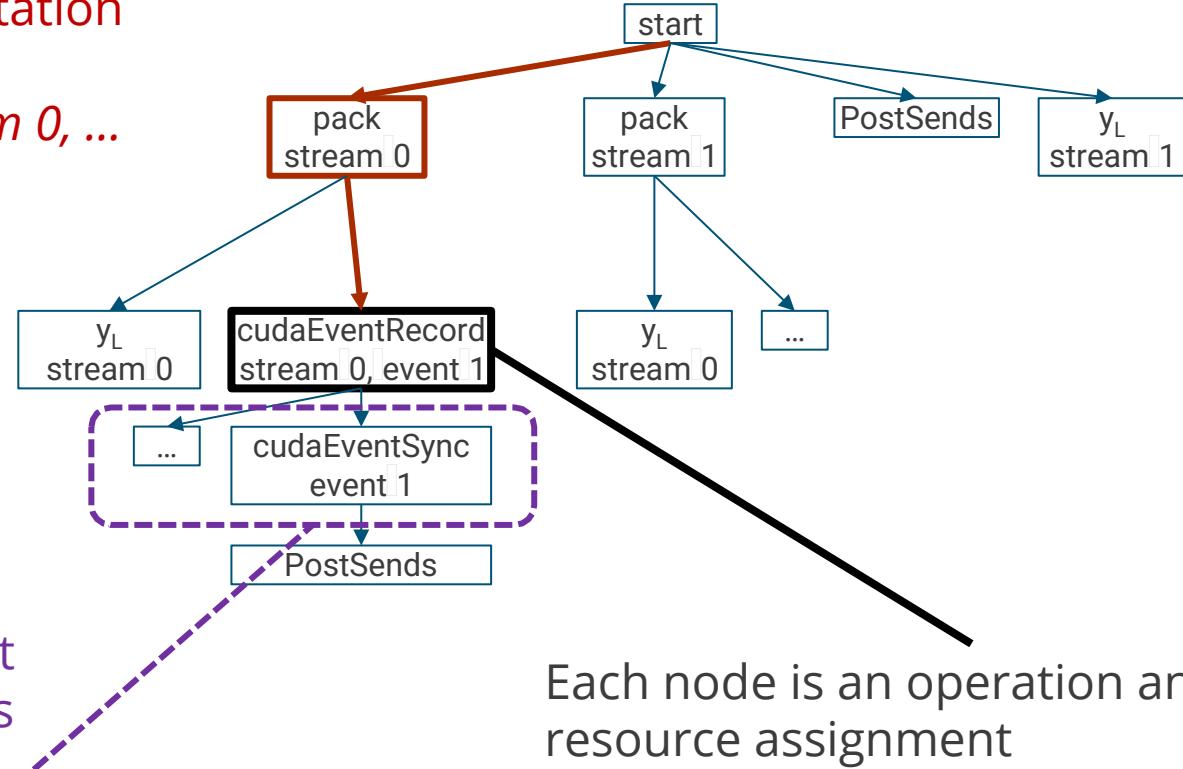
MCTS Represents Search State in a Tree



State space search is stored in a tree

Path is the beginning of an implementation

pack in stream 0, record event 1 in stream 0, ...



Children are all possible subsequent operation / resource combinations

All DAG predecessors complete and synchronized

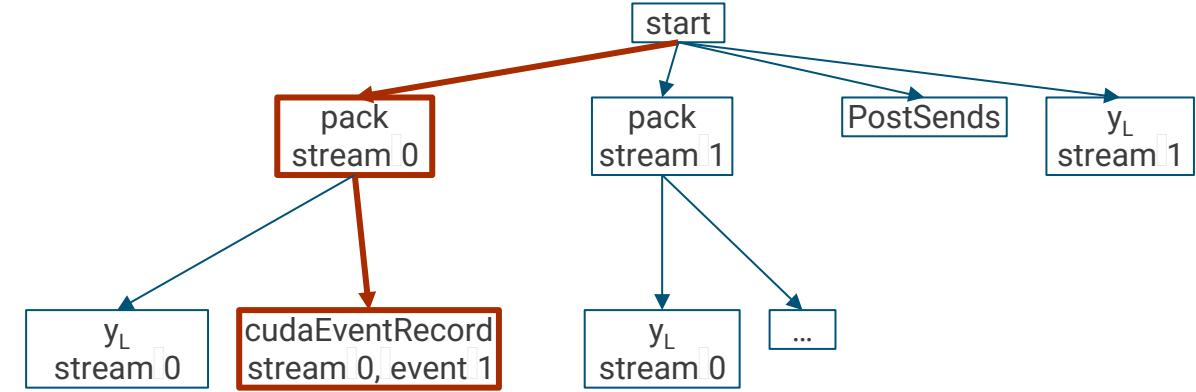
Each node is an operation and resource assignment

From DAG, or synchronization operation

MCTS Iteratively Grows Tree to Focus on Valuable Regions



Selection: Choose a path through the tree

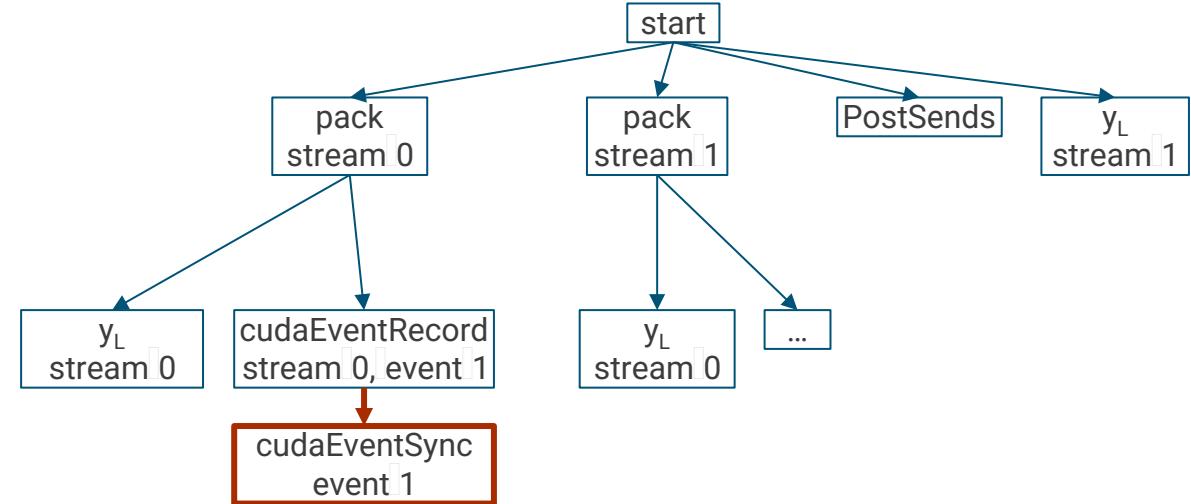


MCTS Iteratively Grows Tree



Selection: Choose a path through the tree

Expansion: Create a new child



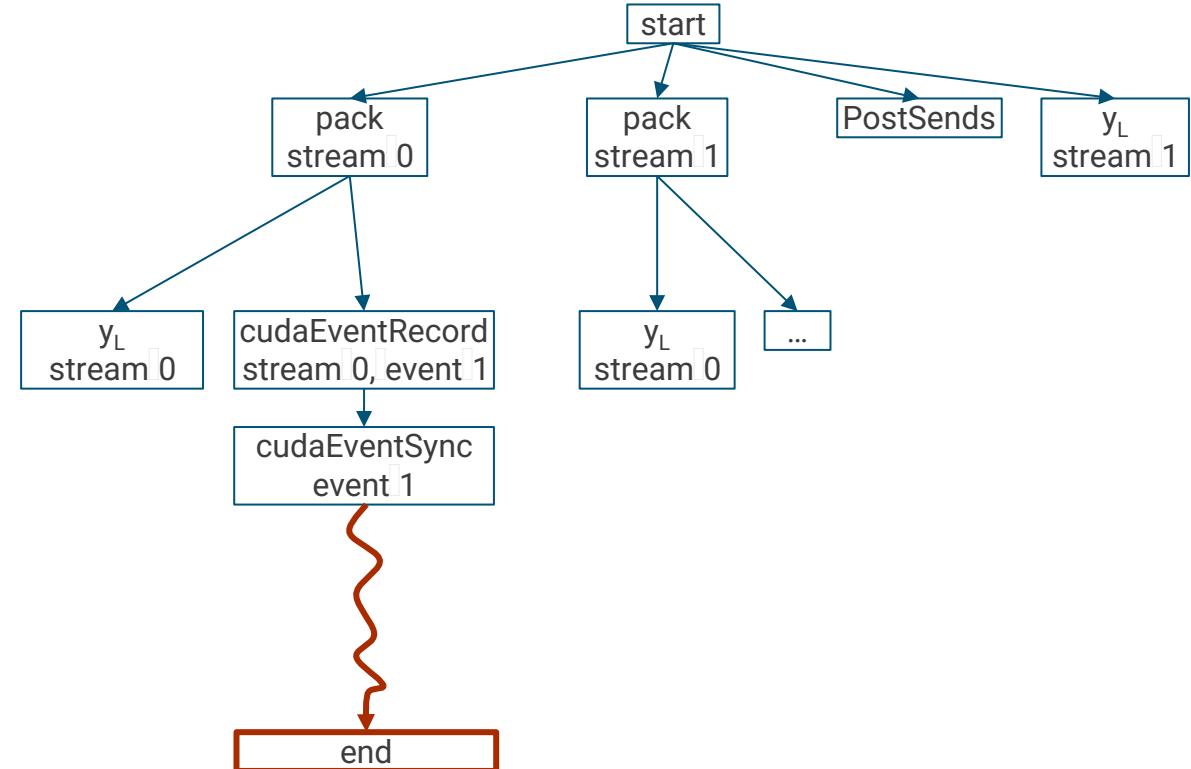
MCTS Iteratively Grows Tree



Selection: Choose a path through the tree

Expansion: Create a new child

Rollout: Random ordering / assignment to complete the implementation



MCTS Iteratively Grows Tree

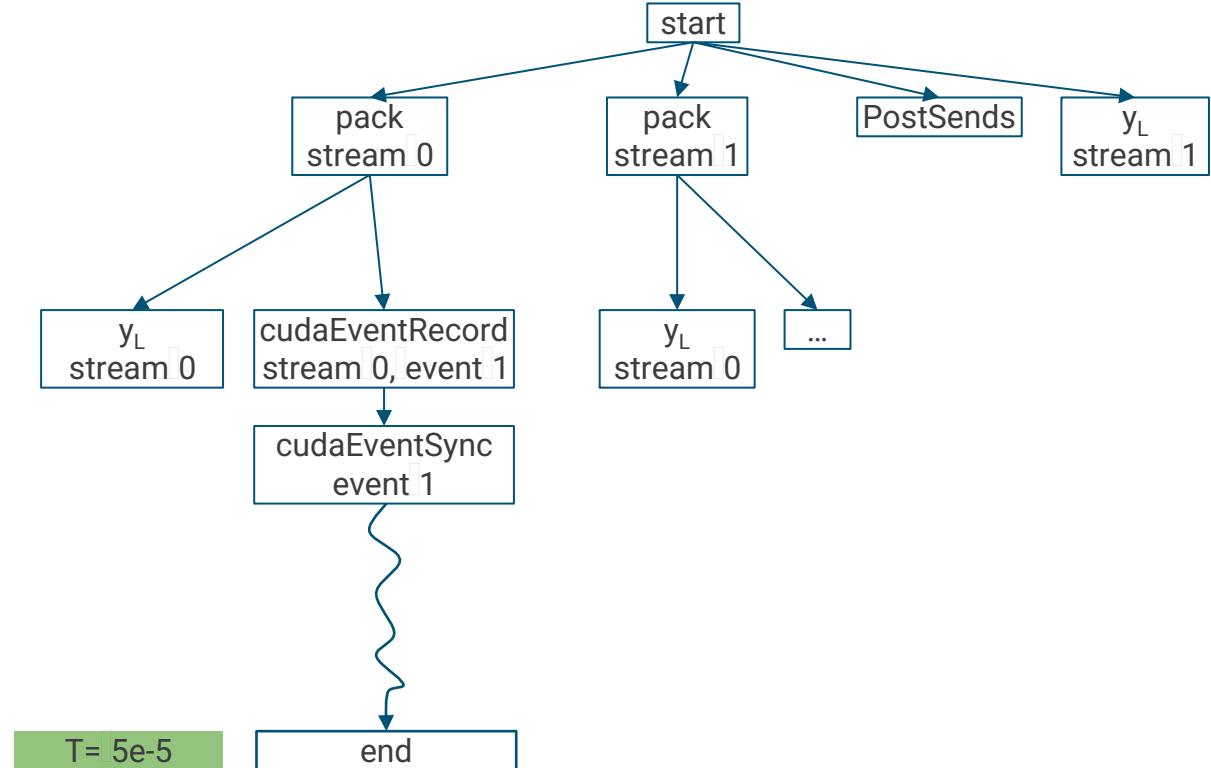


Selection: Choose a path through the tree

Expansion: Create a new child

Rollout: Random ordering / assignment to complete the implementation

Evaluation: Empirical benchmark



MCTS Iteratively Grows Tree



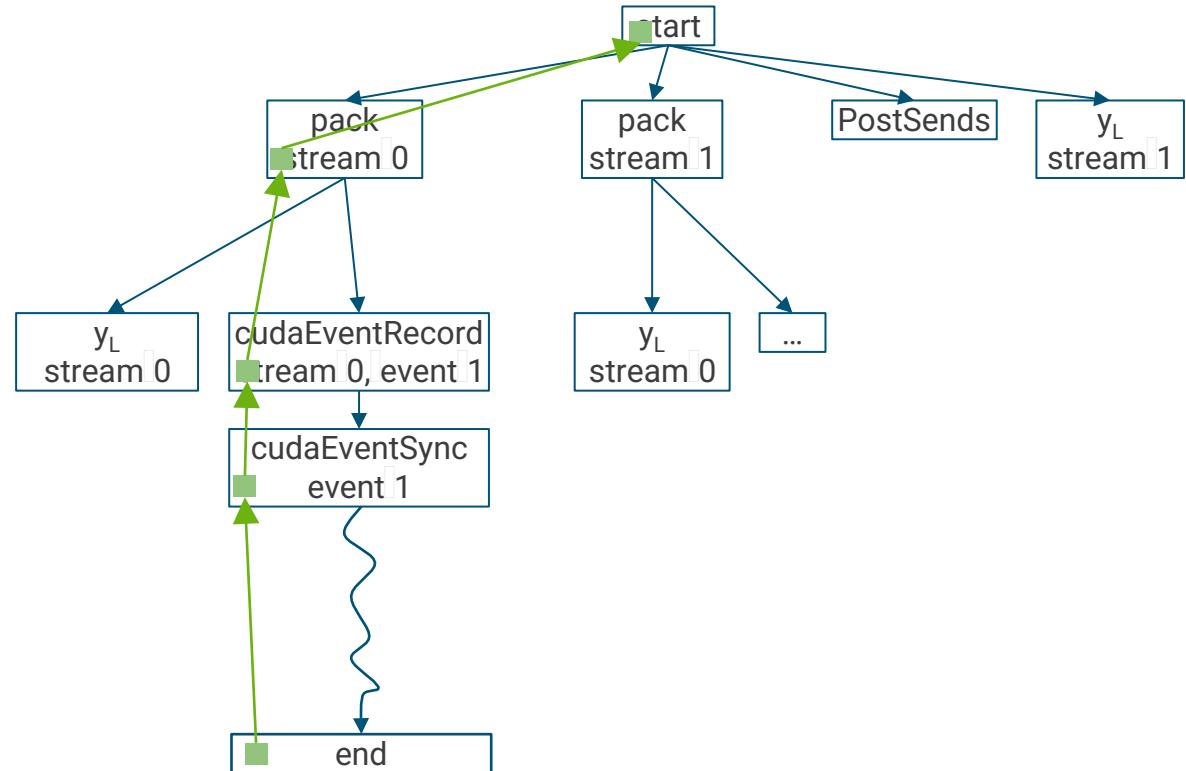
Selection: Choose a path through the tree

Expansion: Create a new child

Rollout: Random ordering / assignment to complete the implementation

Evaluation: Empirical benchmark

Backpropagation: Store result along path

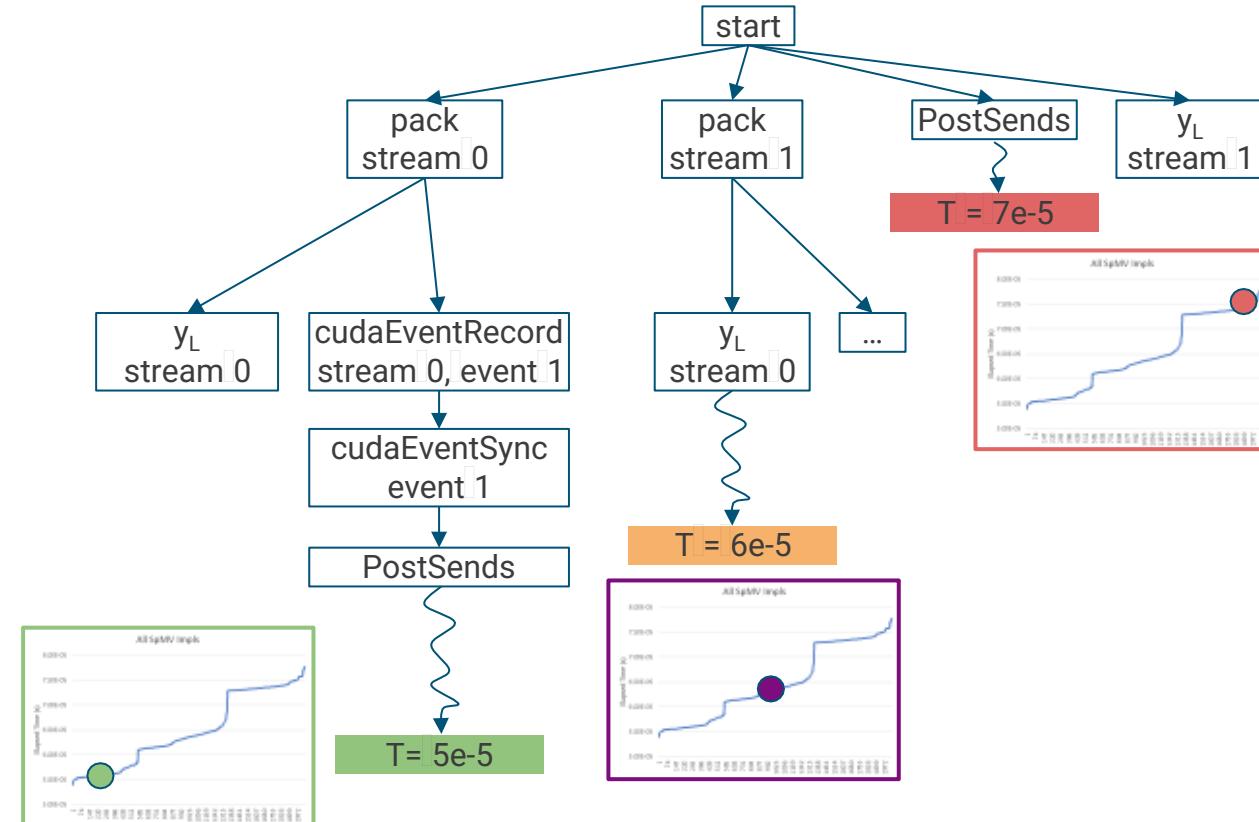


Tree is Deeper and Larger in Valuable Regions

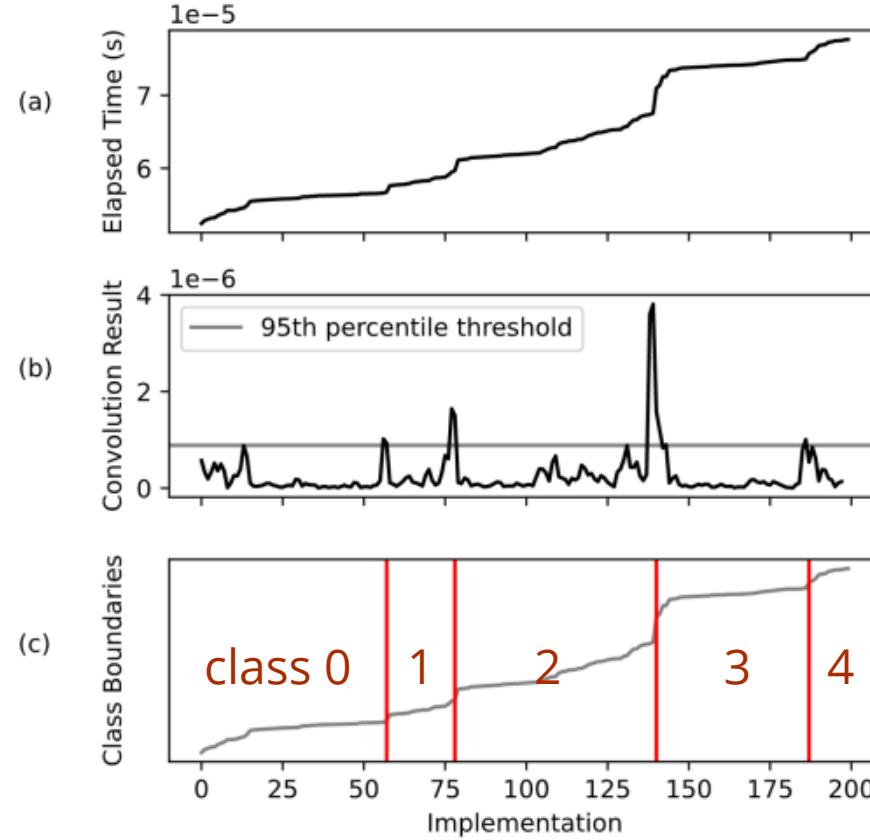


As iterations proceed, tree preferentially explores high-reward regions of the design space

Store all complete implementations and performance results in a table as we go



Transform Empirical Results into Performance Classes and Feature Vectors



subset explored by MCTS



ordering rules *resource assignment rules*

Impl.	Class Label	A then B	...	A same stream B	...
98	2	0	1	1	0
0	0	1	0	1	0
56	1	0	0	1	1
73	1	0	0	1	1
...					...

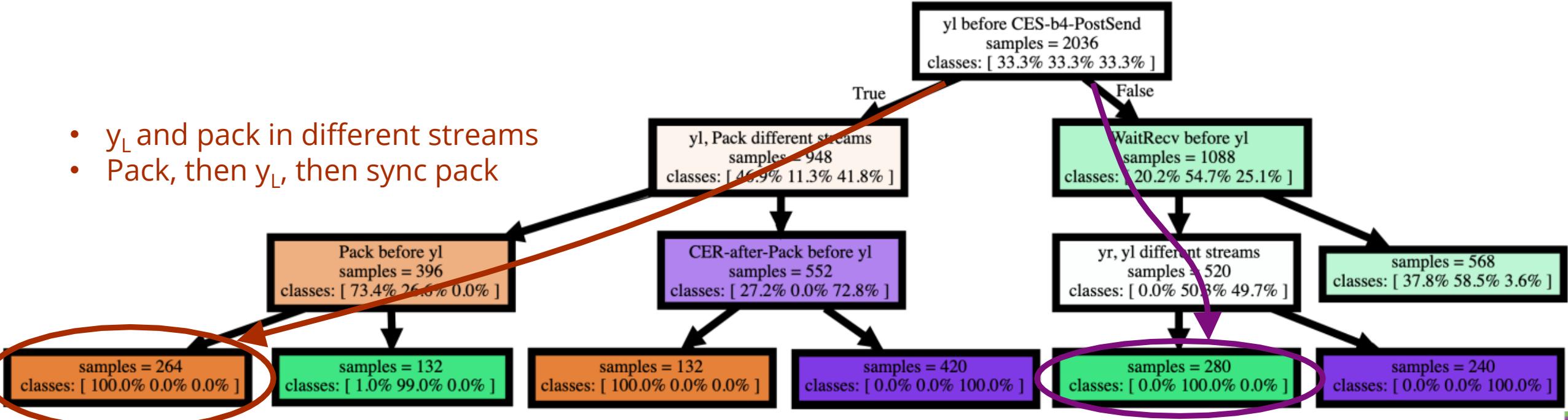
automatic class labeling to identify performance classes
(convolution & peak detection)

feature vectors encode which rules an implementation follows
(sequence-to-vector transformation)

Decision Tree Training to Determine which Rules Discriminate between Classes



- y_L and pack in different streams
- Pack, then y_L , then sync pack



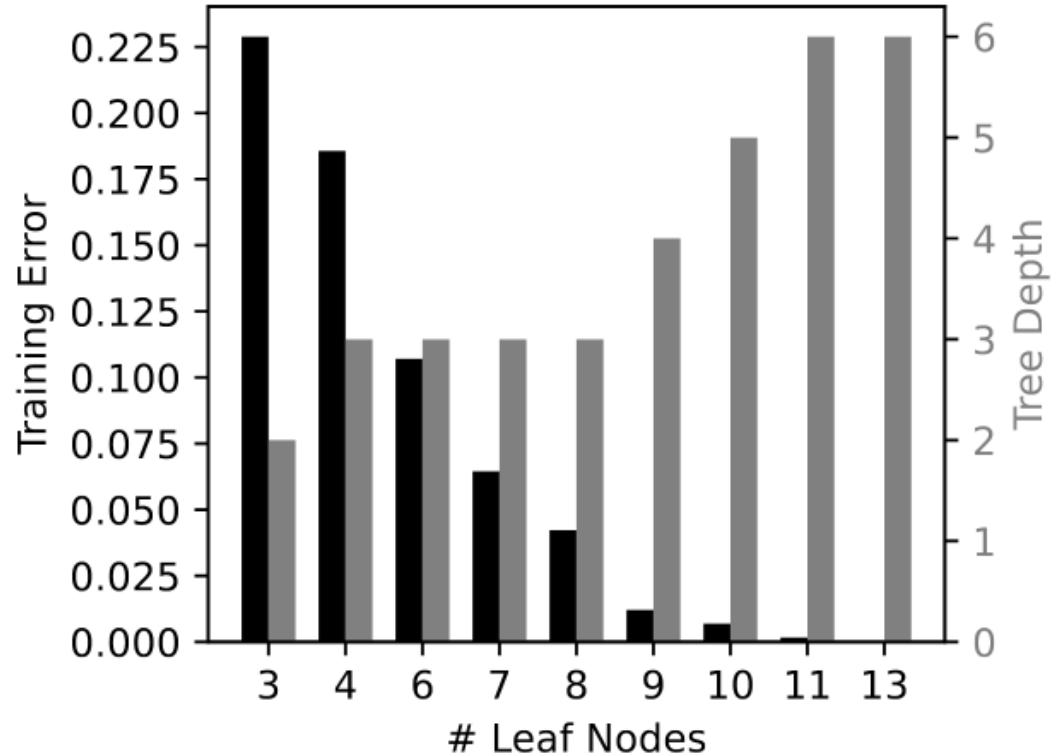
- sync pack before y_L
- WaitRecv before y_L
- y_L, y_R in same stream

Each path through the tree is a set of design rules that define a performance class

Train an Accurate Decision Tree



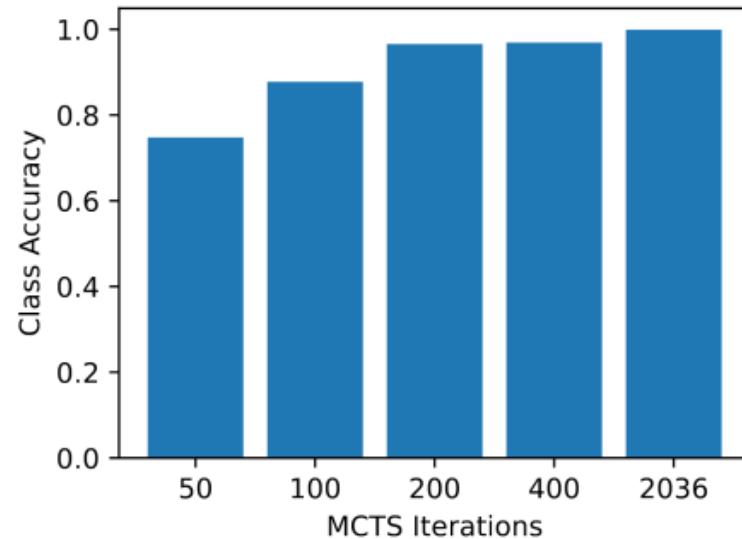
- Training process is for isolating discriminating features
 - **not** for classifying unseen inputs
- Incrementally increase tree size until 100% accuracy achieved
- Accuracy-complexity tradeoff in generated rules



Does MCTS Find Relevant Design Space Regions?



- Each MCTS iteration is a costly empirical benchmark
- Rule quality with reduced iterations?
 - For a given # of iterations, how accurate are the rules?
 - For a given # of iterations, qualitative look at the rules?



MCTS Iterations	2036	50	100	200	400
Discovered Ruleset for Fastest Performance Class	$y_L \rightarrow \text{CES-b4-PostSend}$ $y_L \times \text{Pack}$ $\text{Pack} \rightarrow y_L$	$y_L \rightarrow \text{CES-b4-PostSend}$ $y_L \times \text{Pack}$ $\text{Pack} \rightarrow y_L$	$y_L \rightarrow \text{CES-b4-PostSend}$ $y_L \times \text{Pack}$ $\text{Pack} \rightarrow y_L$ $y_L \rightarrow \text{WaitSend}$	$y_L \rightarrow \text{CES-b4-PostSend}$ $y_L \times \text{Pack}$ $\text{Pack before } y_L$ $y_L \rightarrow \text{WaitSend}$	$y_L \rightarrow \text{WaitRecv}$ $\text{PostSend} \rightarrow y_L$ $\text{Pack} \rightarrow y_L$ $\text{CER-after-Pack} \rightarrow y_L$ $y_L \rightarrow \text{WaitSend}$ $\text{PostRecv} \rightarrow \text{CES-b4-PostSend}$

$A \times B$: A different stream than B

$A \rightarrow B$: A, then B

Most populous ruleset shown

Vision for this work



- Current
 - C++ MCTS implementation for MPI/CUDA codes with multiple streams
 - Prototype feature-vector and decision tree training using SciKit in Python
 - In progress open sourcing
- Upcoming
 - Apply to key Tpetra distributed linear algebra operations
 - MCTS search strategies
 - Better rollout techniques
- Future Explorations
 - Identify unexpected performance effects on target platforms (“performance bugs”)
 - What to do as communication / computation are more tightly integrated
- Summary
 - Represent CUDA+MPI operation as DAG
 - Automatically generate human-interpretable rules for library design
 - Maintain human provenance of implementation (no “black boxes”)

About the Sandia Brand Templates



Create impactful presentations, reports, and visuals with Sandia branded PowerPoint templates.

FEATURES

- 16:9 HD widescreen format
- Embedded [Sandia font, colors](#), and [MS Office Theme](#)
- Professional photo and text layouts available in the [Sample Layouts deck](#)
- Access fully editable charts, maps, and icons in the [PowerPoint Graphics Library](#)
- Easy to use placeholders crop photos without distortion
Note: To reduce file sizes, templates do not include Images. UUR photos can be accessed at Sandia's [Flickr](#) page.
- Before submitting to Sandia Review and Approval, ensure only the appropriate markings are applied to content slides and Slide Masters.

Questions?

Get immediate support from Creative Services.

NM: (505) 844-7167 | **CA:** (925) 294-1010 | creative.sandia.gov

Want more?

Browse additional design templates at
creative.sandia.gov

Revised 06.18.21