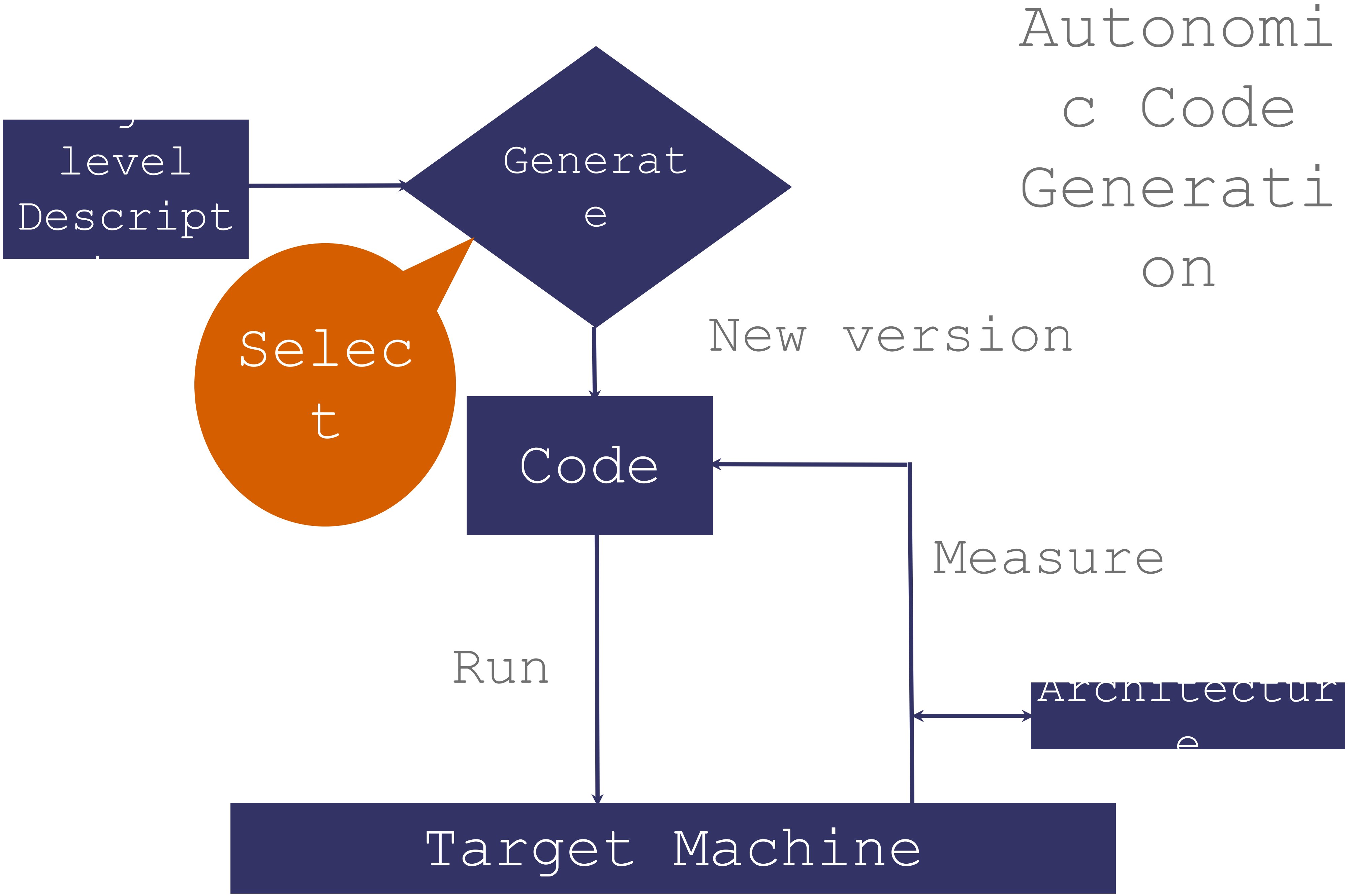
Optimizing Sorting with Machine Learning Algorithms

Xiaoming Li*, María Jesús Garzarán, and David Padua
University of Illinois at Urbana-Champaign

Autonomic code generation

- Automatically produces efficient implementations for a wide range of platforms
- Related works
  - PhiPAC (Berkeley), ATLAS (Tennessee)
    - Basic Linear Algebra Routines (BLAS)
  - Spiral (CMU), FFTW (MIT)
    - Signal Processing Algorithms

Autonomic Code Generation



Opportunities for improvement

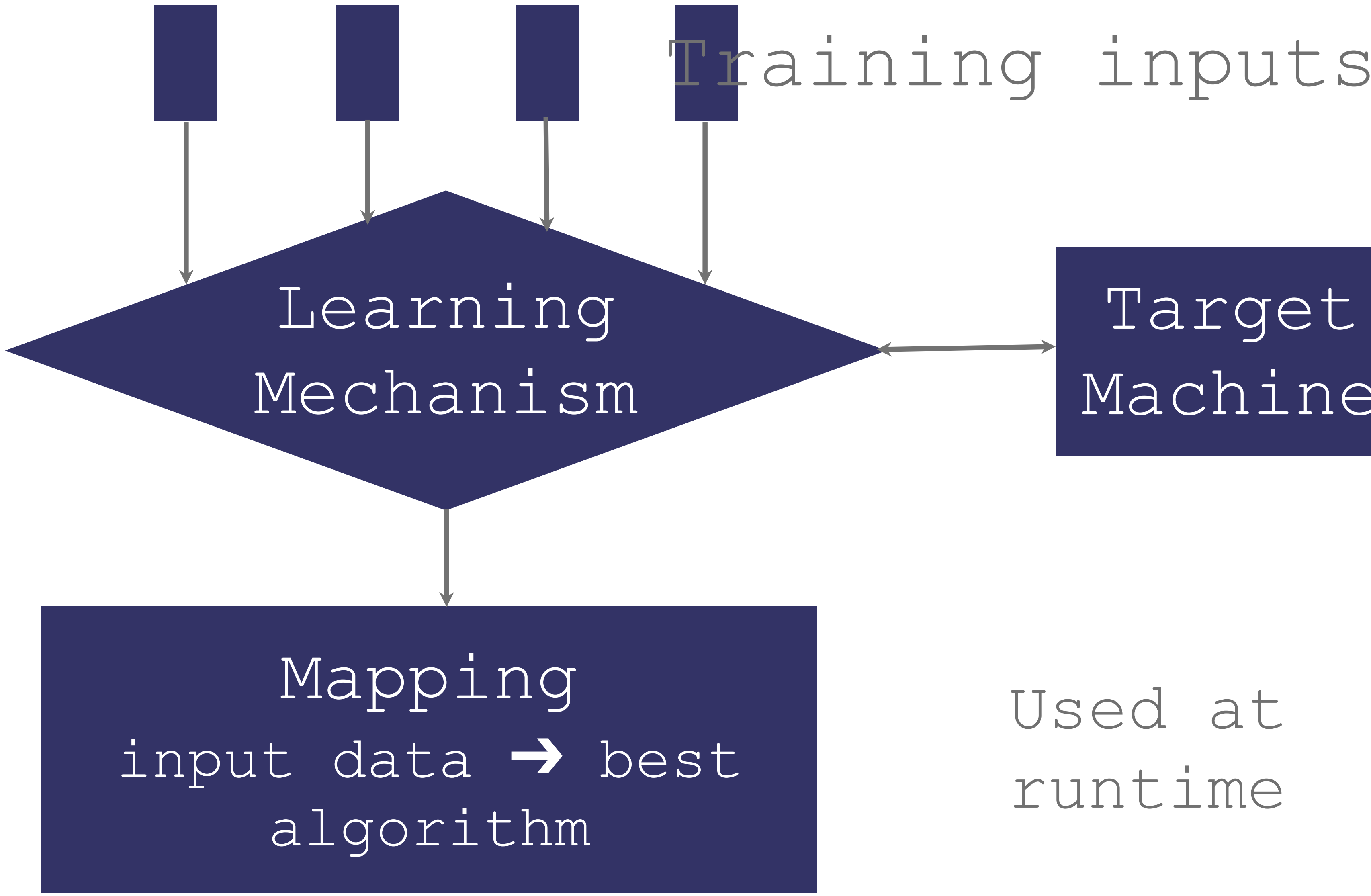
- Adapt to input characteristics
  - When the performance depends on inputs

Contributions of this project

- Apply machine learning techniques to generate code that adapts to input data characteristics
  - At runtime, select one of a few algorithms
  - Combine algorithms to generate new algorithms.

How to generate efficient sorting routines?

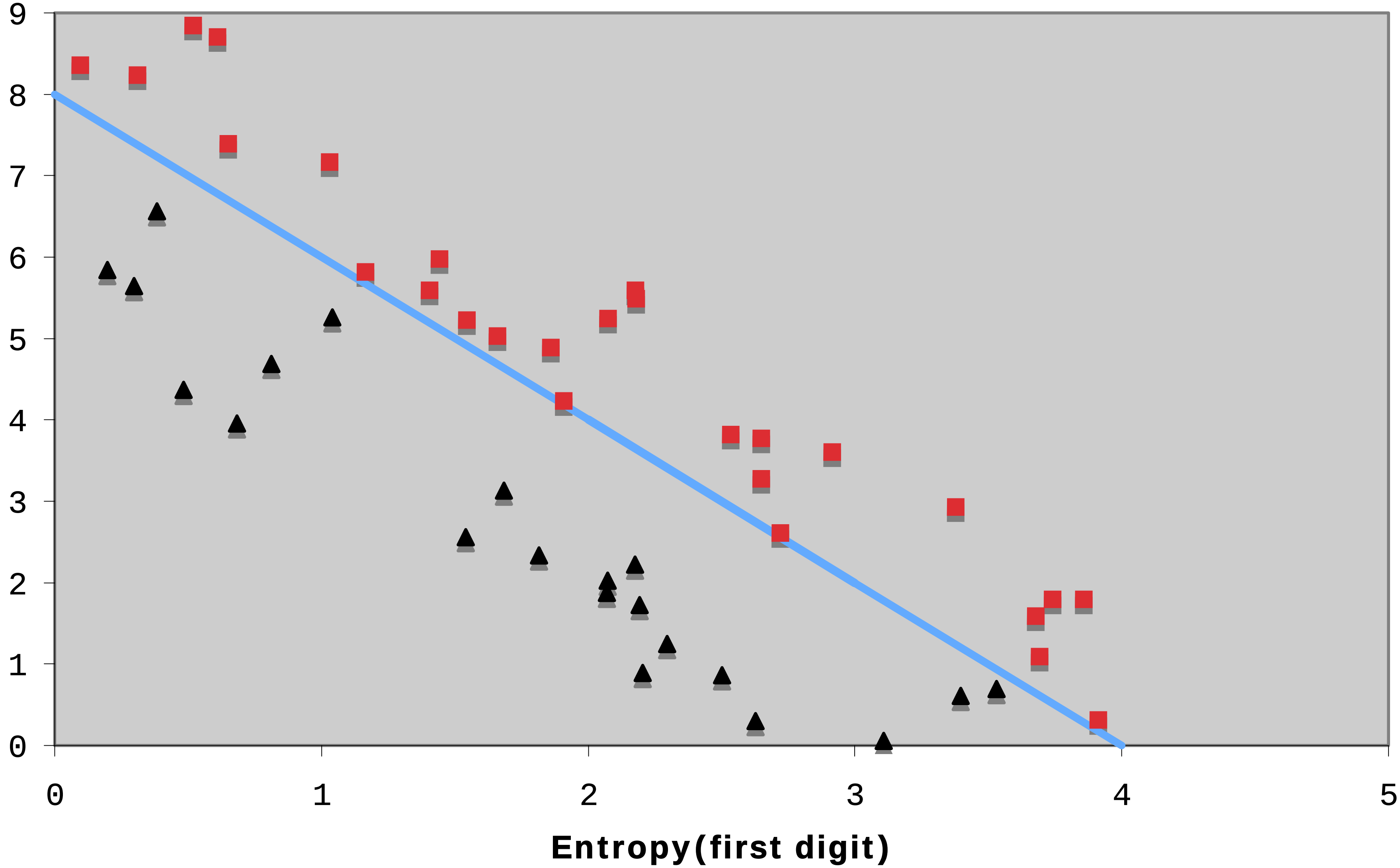
Selection of the best sorting routine



Sorting routine candidates

- Quicksort
- Multi-way Merge Sort
- Radix Sort

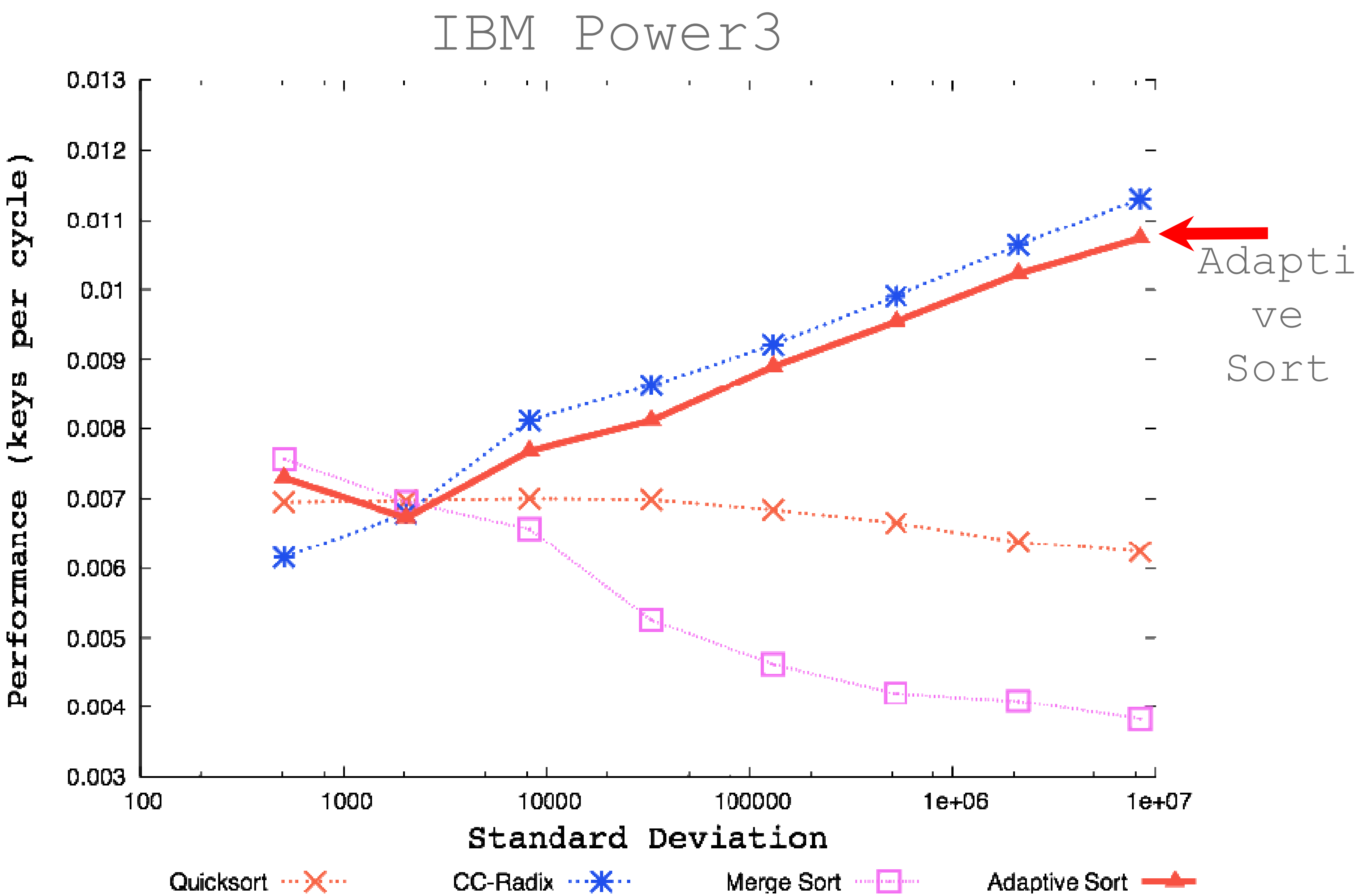
Learn linear separable function



Experiment platforms

- IBM Power3
- IBM Power4
- Intel Itanium 2
- Intel Xeon
- Sun UltraSparcIII
- SGI R12k
- AMD Athlon MP

Results on IBM Power3



Generate efficient hybrid code

- Abstract basic operations
  - sorting primitives
- Build hybrid sorting routines from primitives
  - Adapt to architectural features and input characteristics

Abstract sorting primitives

- Partitioning methods
  - Divide-with-pivot (DP)
  - Divide-into-block (DB)
  - Divide-by-digit-from-left (DR)
  - Divide-by-digit-in-middle (DRU)

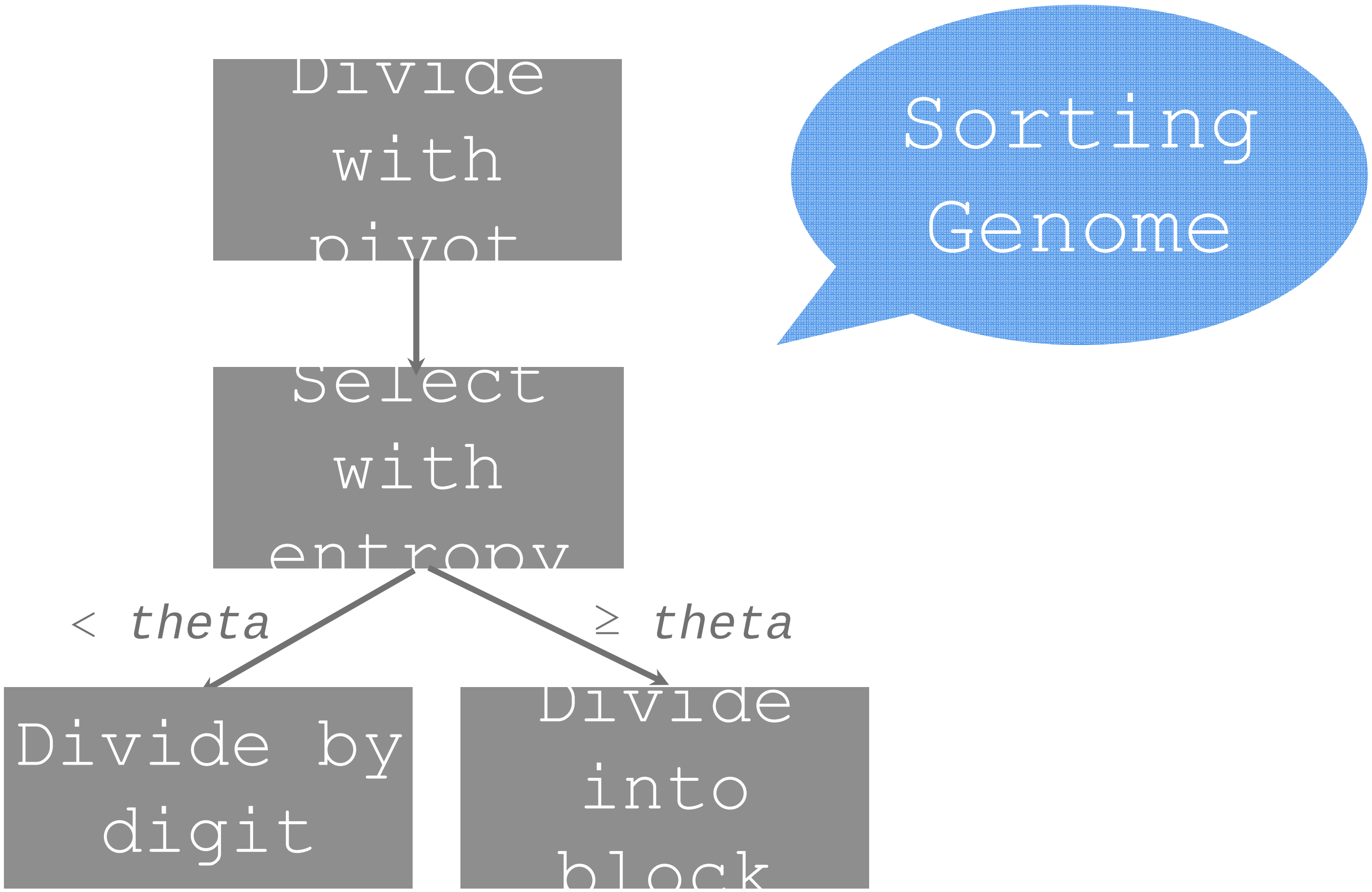
From Quicksort

From Merge Sort

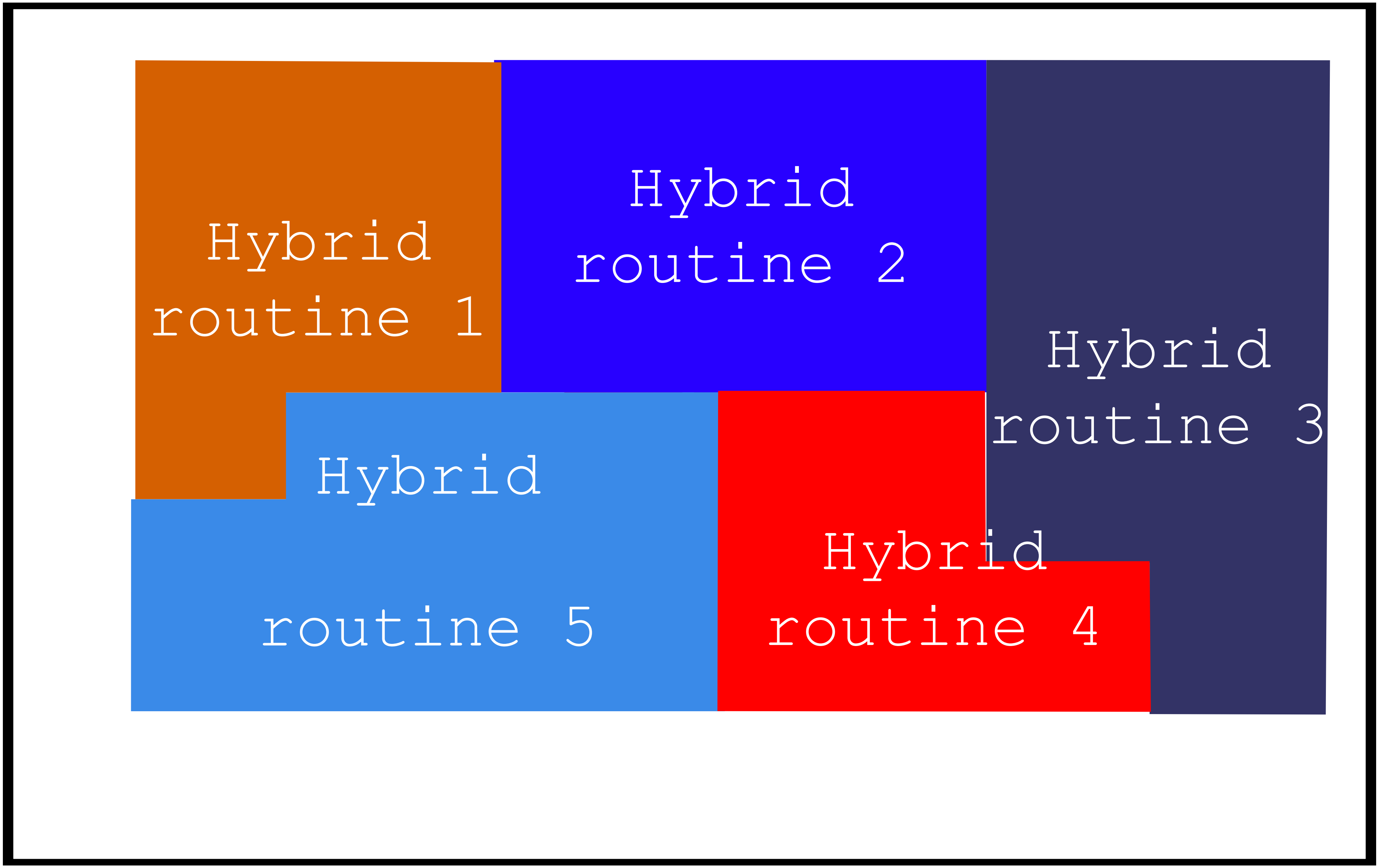
From Radix Sort

- How to choose a partitioning method
  - Using the size of the partition (BN)
  - Using the entropy of the partition (BE)

Example of hybrid sorting



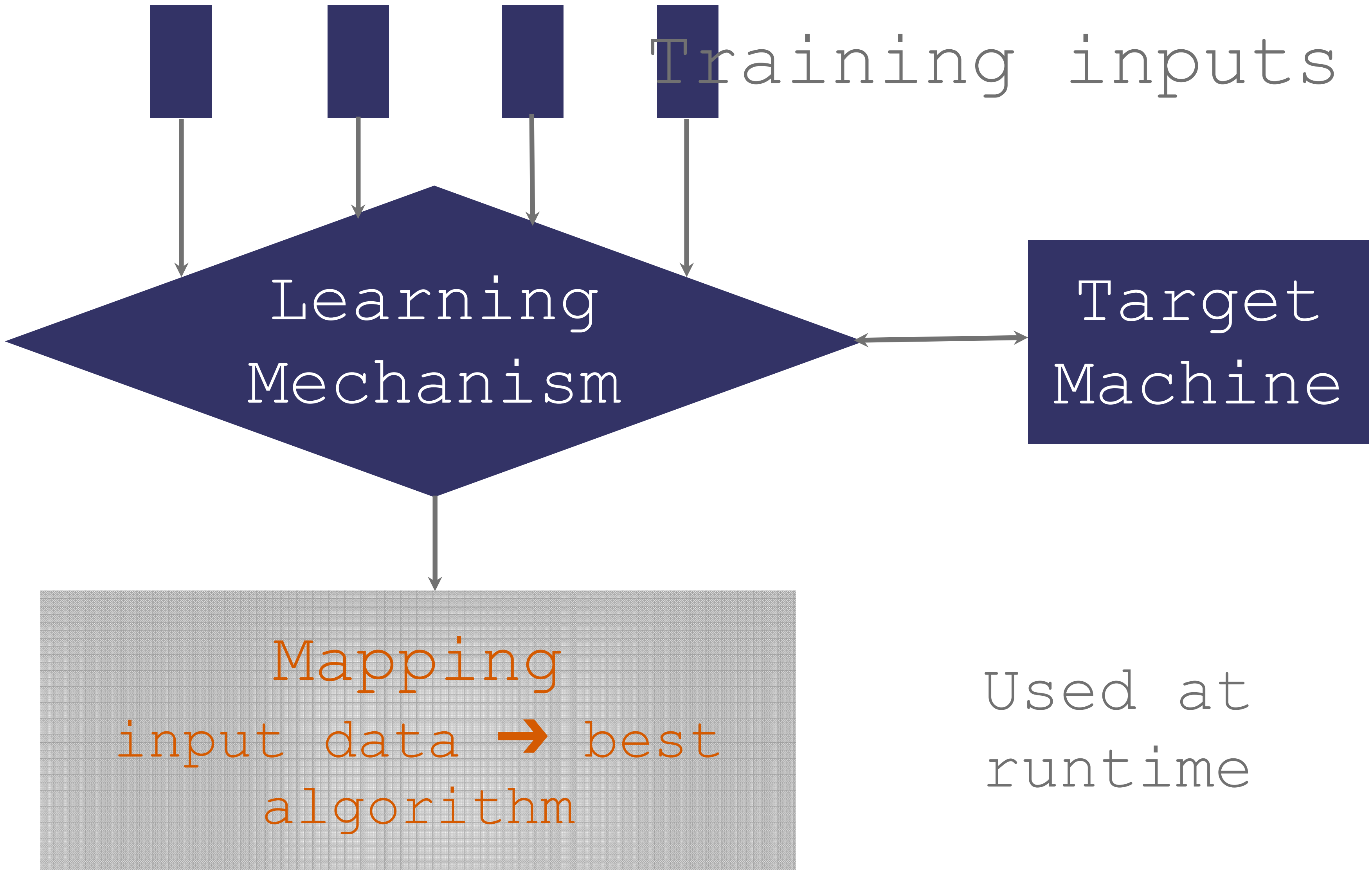
Hybrid algorithms complicate partition



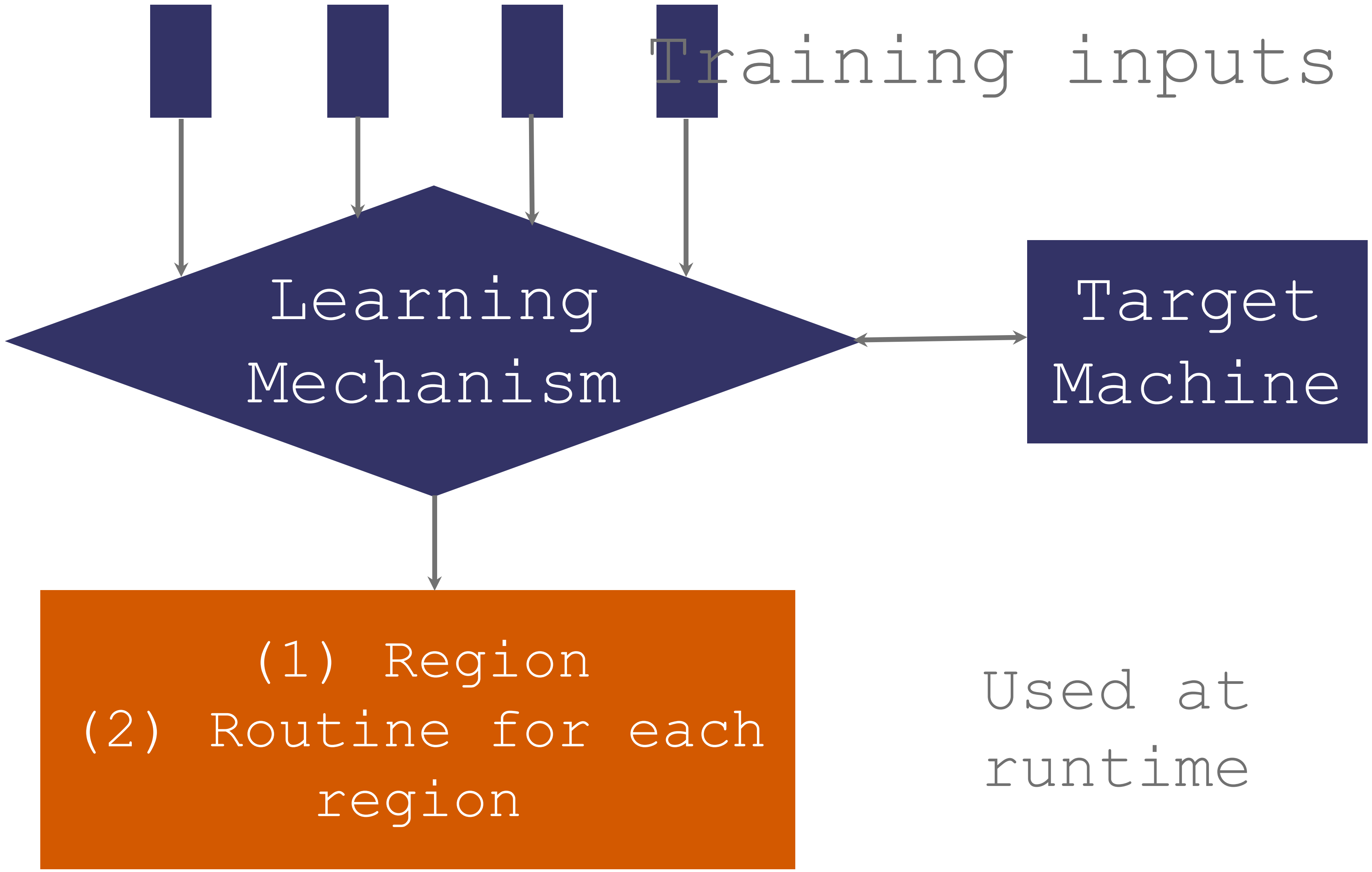
Build the best sorting routine

- Challenges
  - Huge number of possible sorting routines
  - Adapt to architectures and inputs in regions
- Use machine learning algorithms to guide the synthesis
  - XCS, a Learning Classifier System

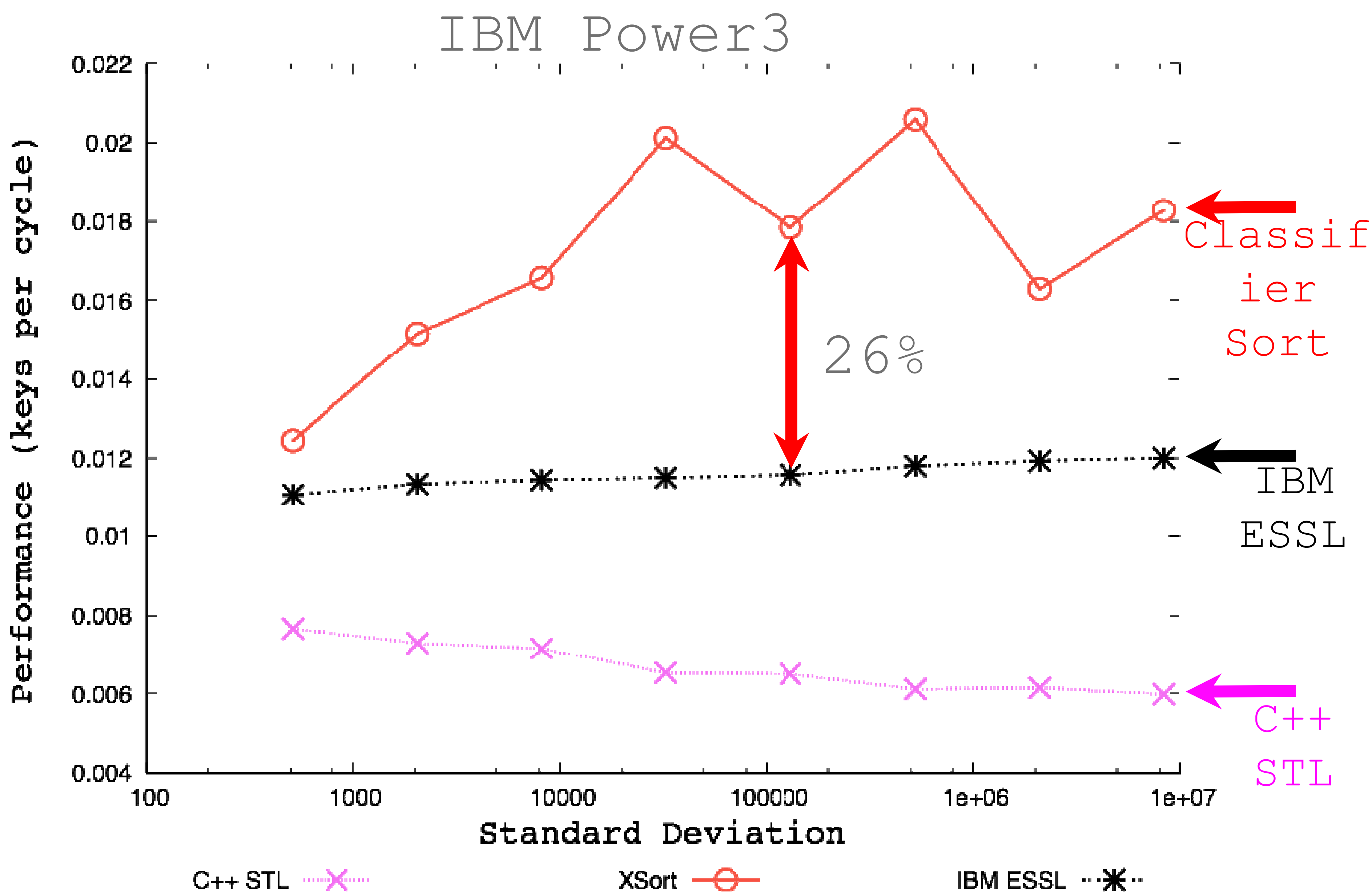
Synthesize hybrid sorting routines



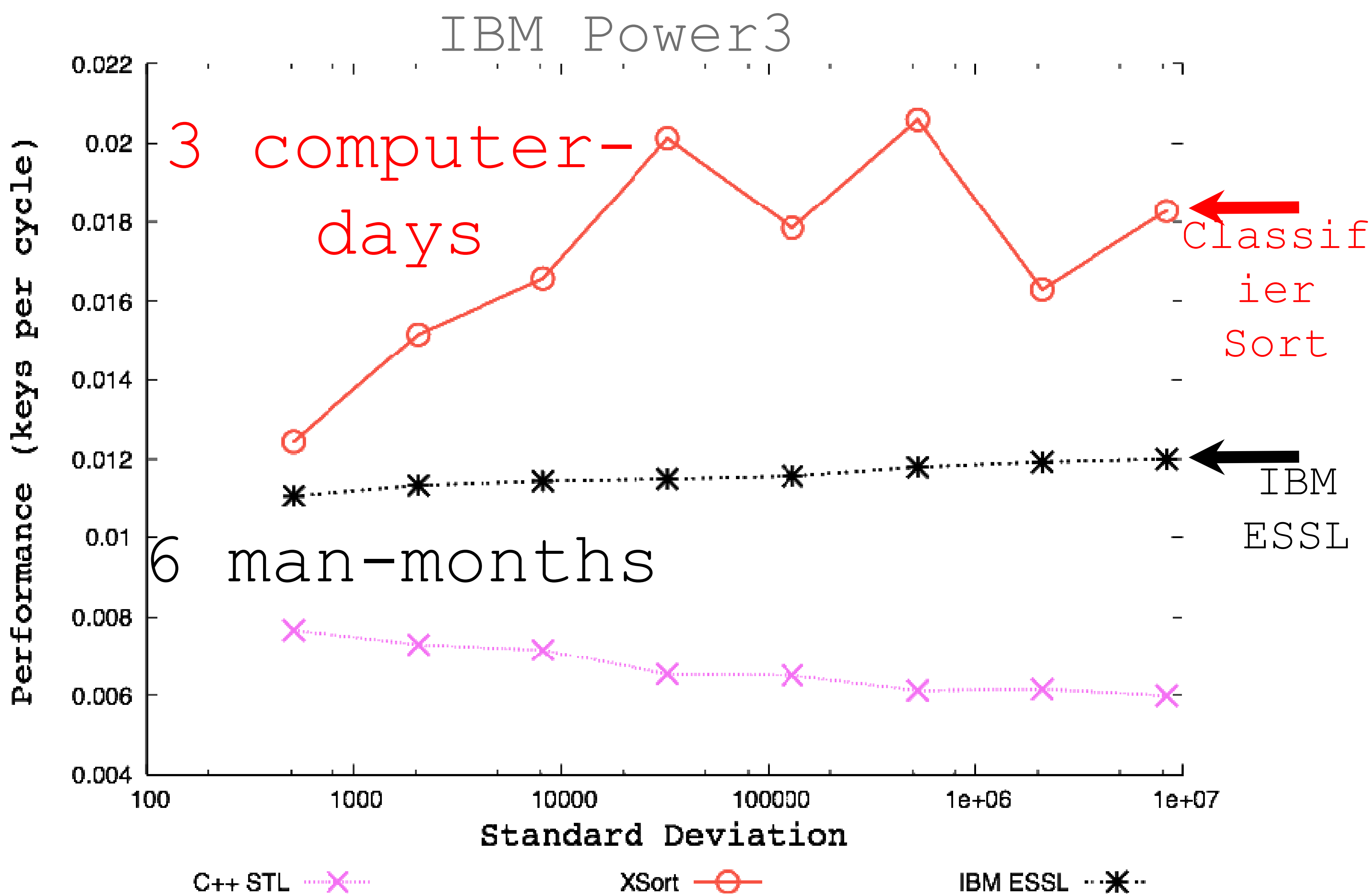
Synthesize hybrid sorting routines



Performance

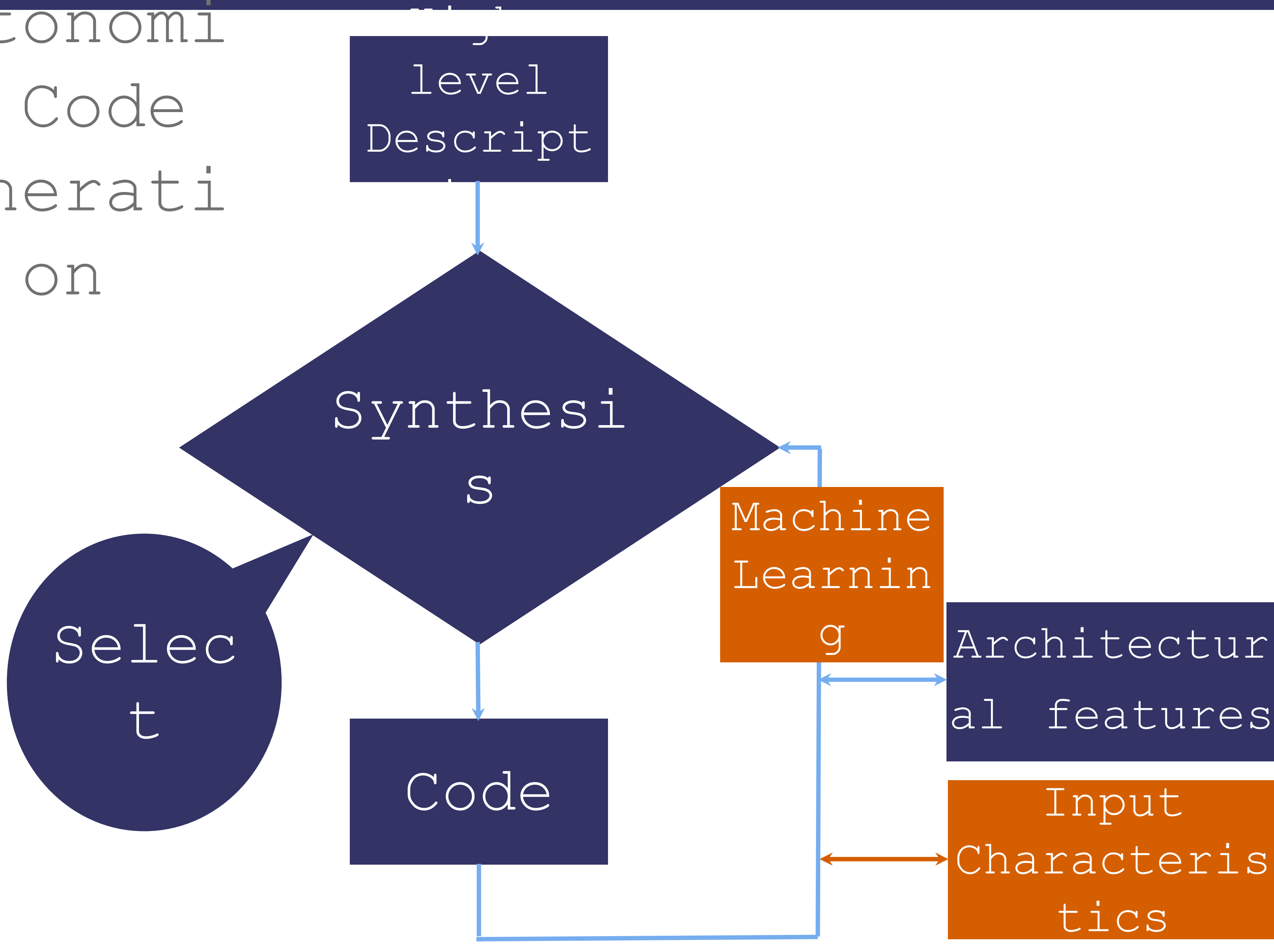


Performance



Summary and future work

Autonomic Code Generation



Summary and future work

- Predict and select the best "pure" sorting algorithm at runtime
  - Accurate prediction with low overhead (~5%)
- Automatically generate hybrid sorting algorithms that outperform all vendor libraries
  - > 20% faster than IBM ESSL using 2% of time