Benchmark Design for Robust Profile-Directed Optimization

SPEC Workshop 2007

Paul Berube and José Nelson Amaral
University of Alberta

NSERC Alberta Ingenuity iCore

In this talk

- SPEC: SPEC CPU
- PDF: Offline, profile-guided optimization
- Test: Evaluate
- Data/Inputs: Program input data

PDF in Research

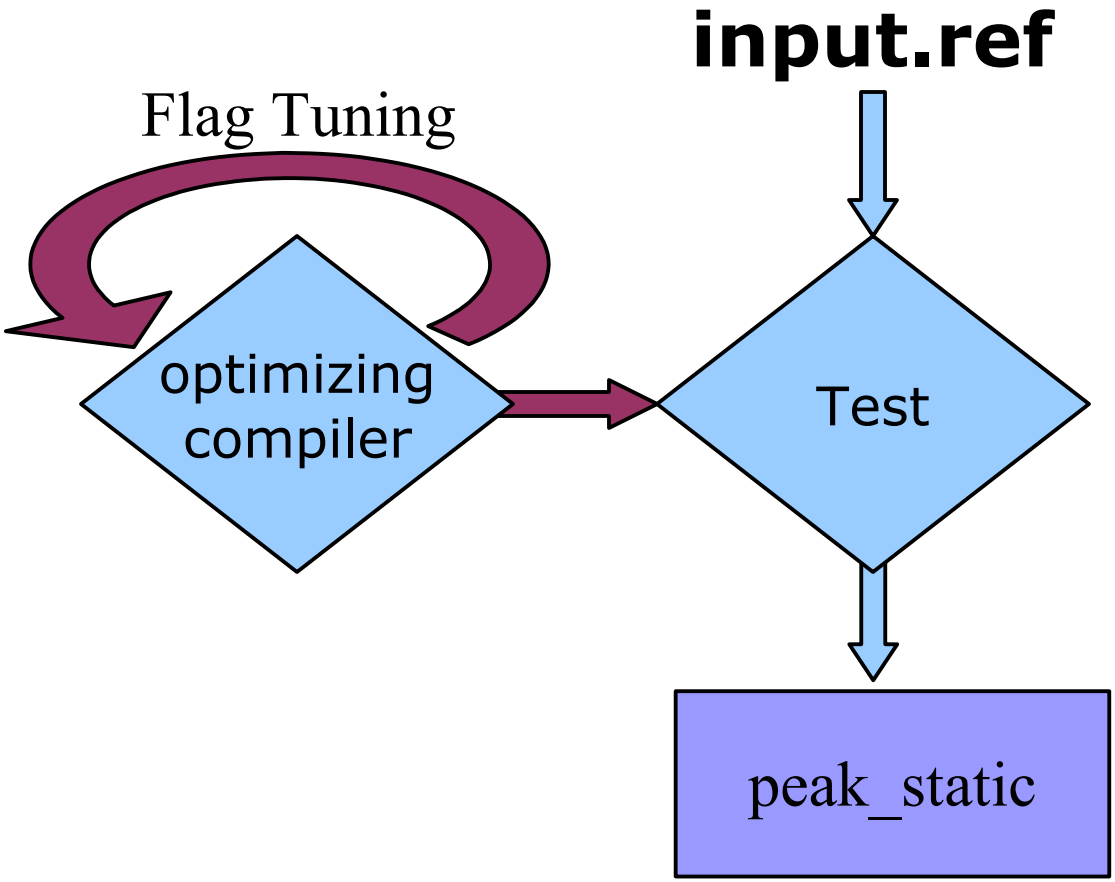
- SPEC benchmarks and inputs used, but rules seldom followed exactly
  - PDF will continue regardless of admissibility in reported results
- Some degree of profiling is taken as a given in many recent compiler and architecture works

An Opportunity to Improve

- No PDF for base in CPU2006
  - An opportunity to step back and consider
- Current evaluation methodology for PDF is not rigorous
  - Dictated by inputs/rules provided in SPEC CPU
  - Usually followed when reporting PDF research

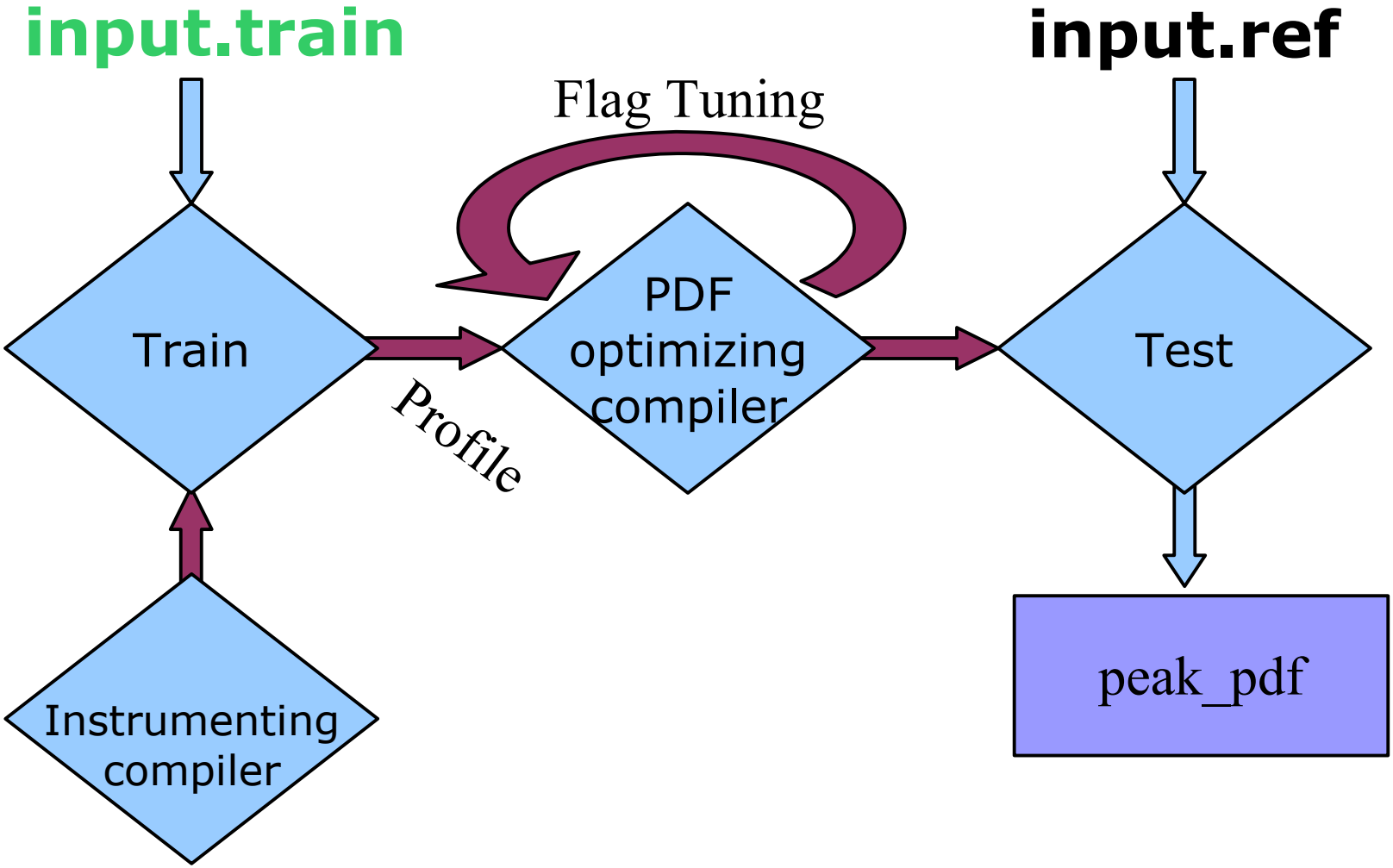
Current Methodology

Static optimization



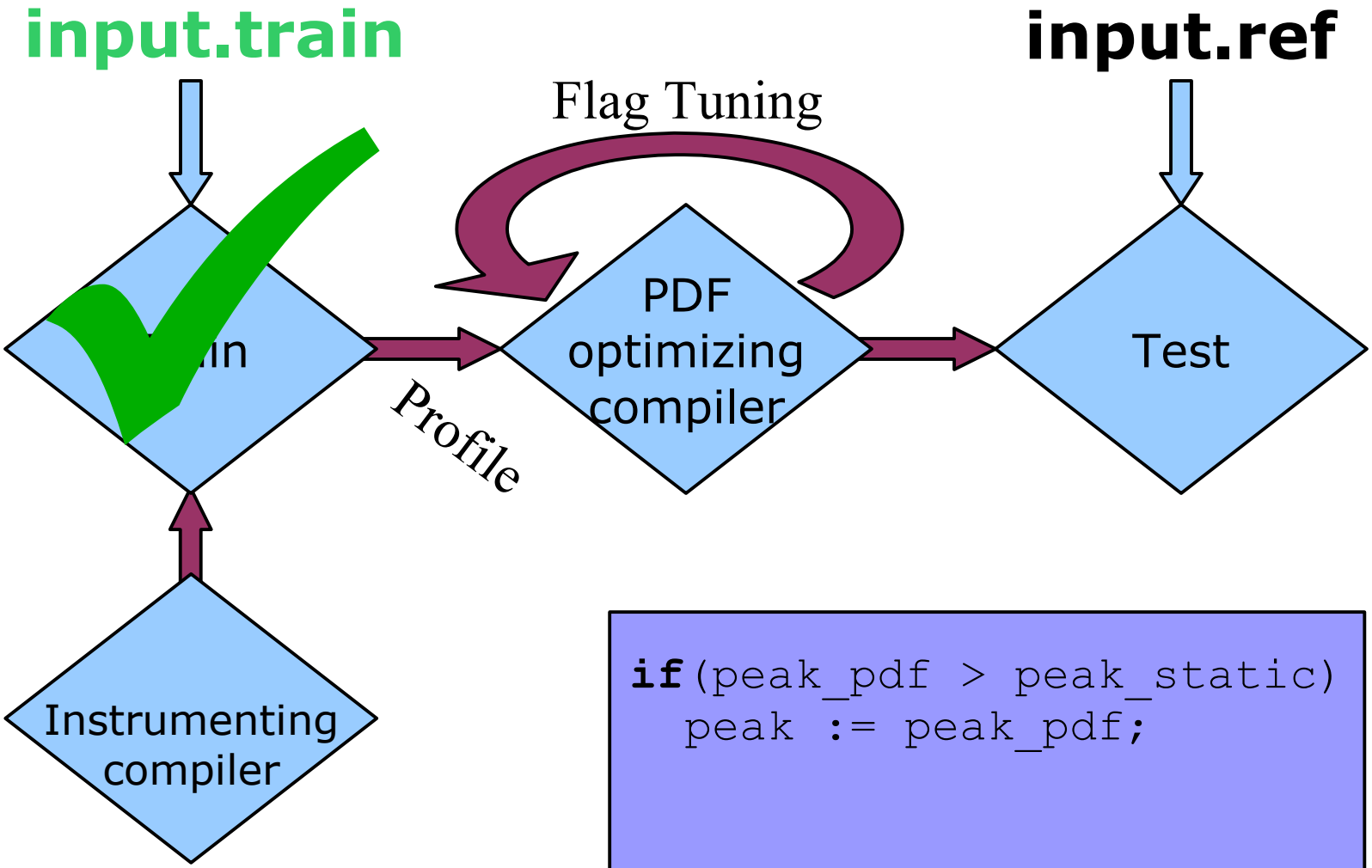
Current Methodology

PDF optimization



Current Methodology

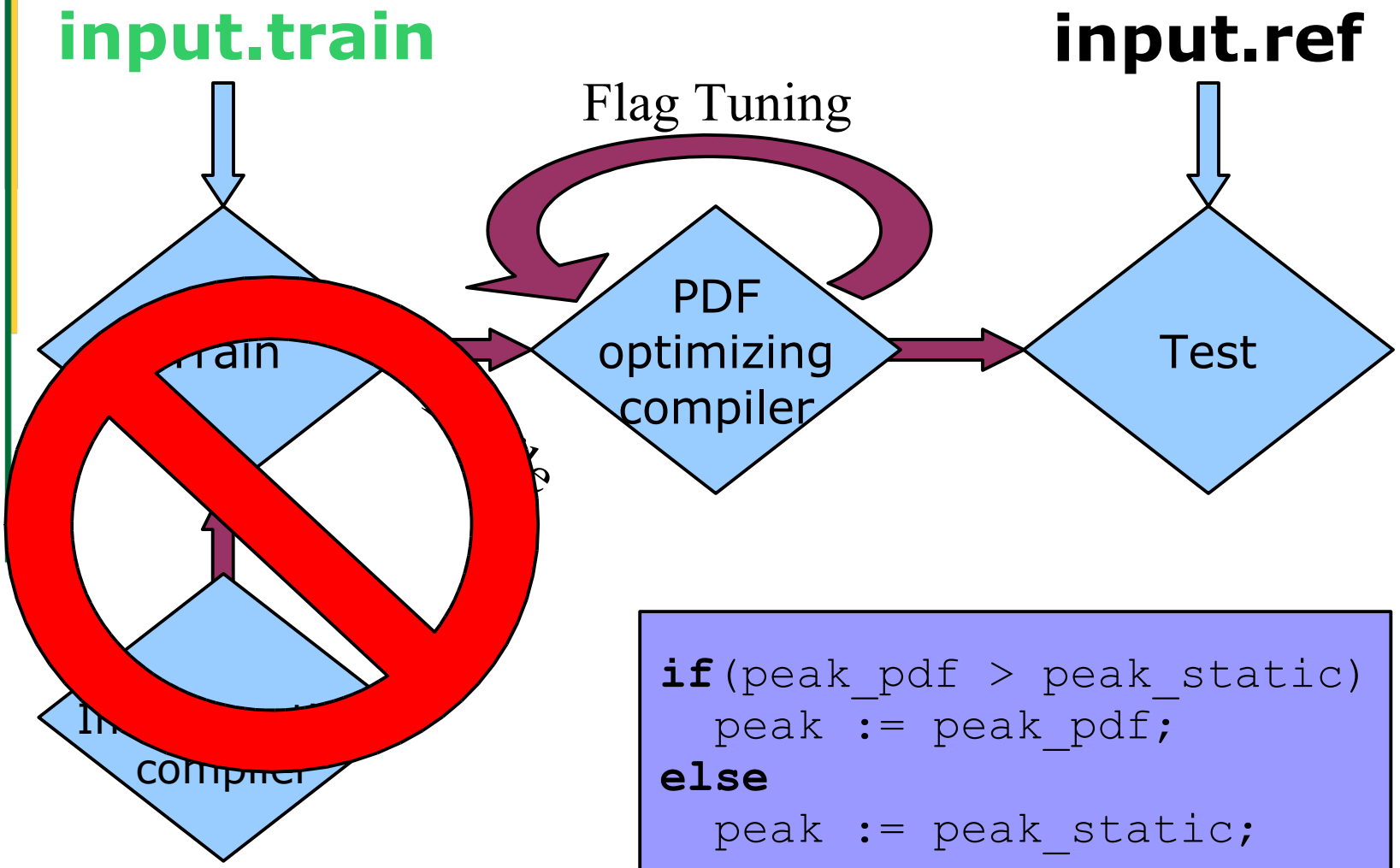
PDF optimization




```
if(peak_pdf > peak_static)
  peak := peak_pdf;
```

Current Methodology

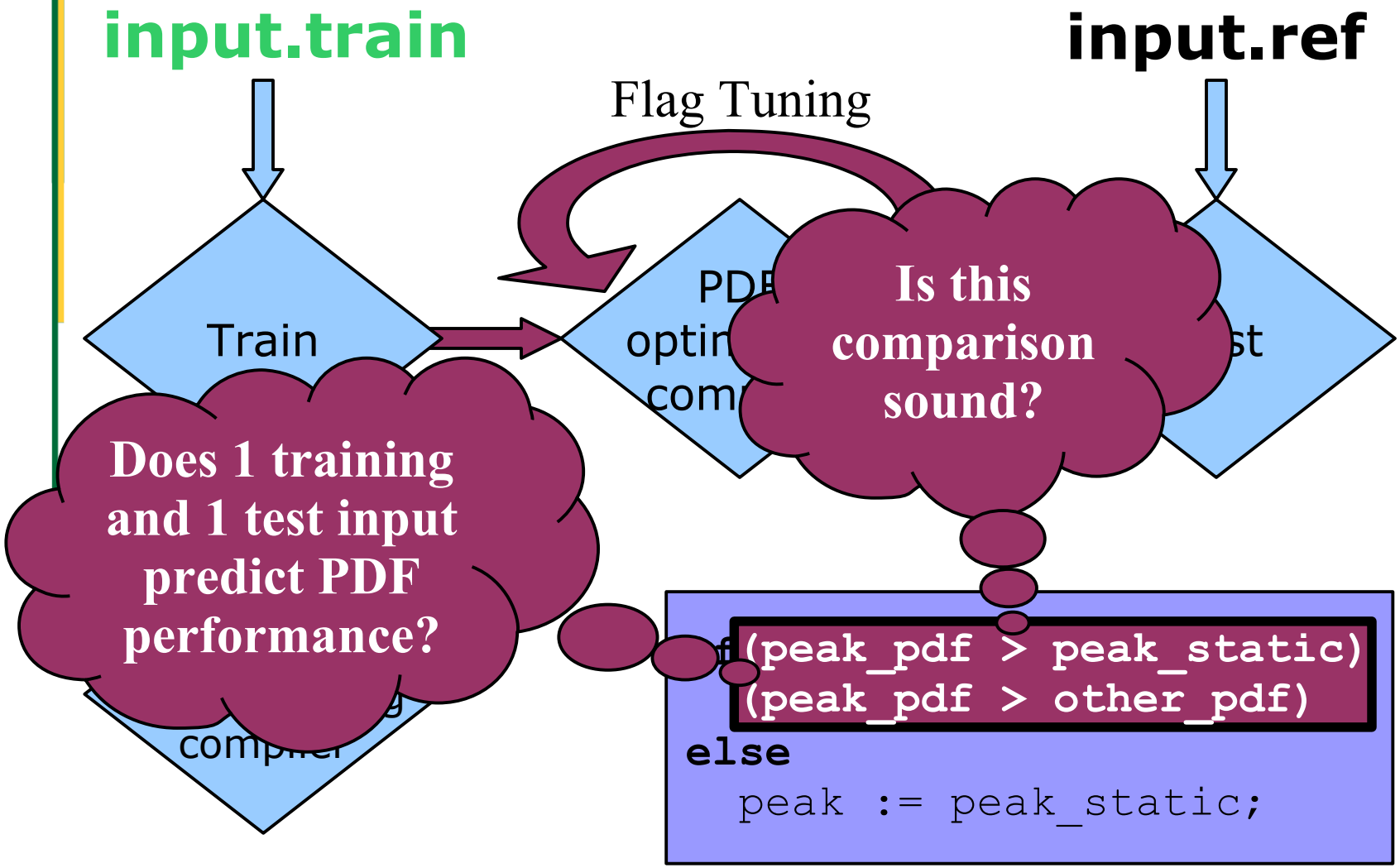
PDF optimization




```
if(peak_pdf > peak_static)
  peak := peak_pdf;
else
  peak := peak_static;
```

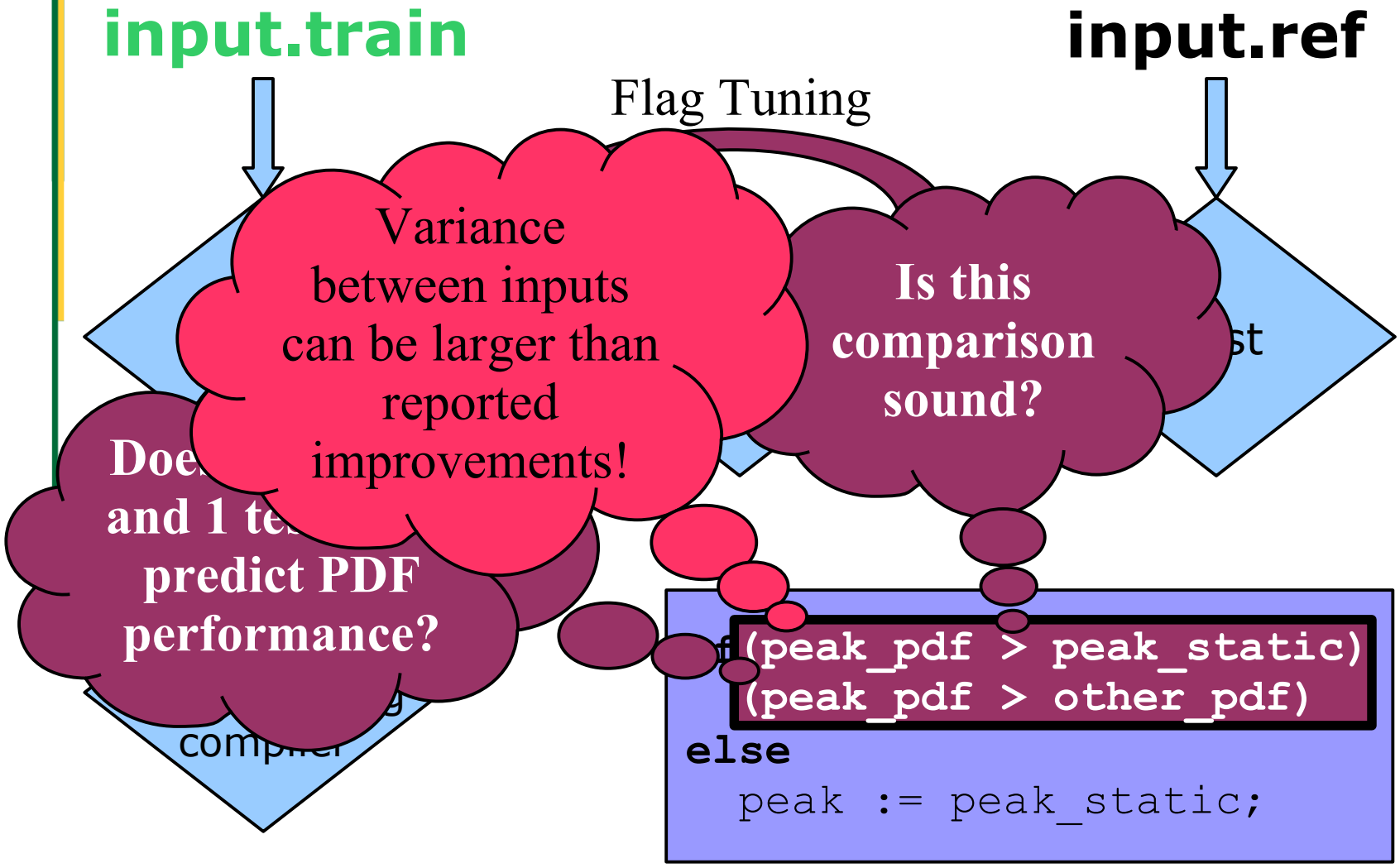
Current Methodology

PDF optimization

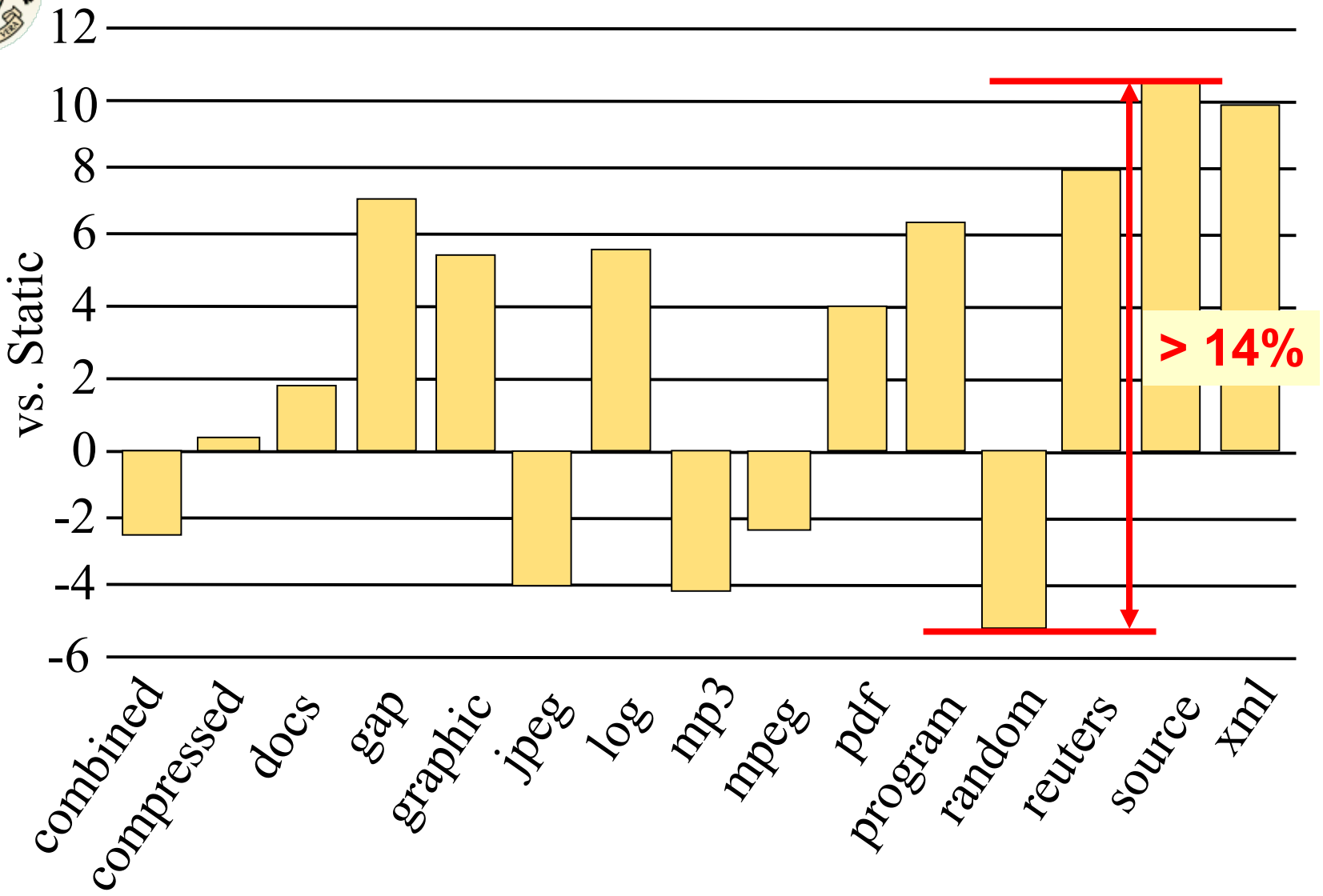


Current Methodology

PDF optimization



bzip2 – Train on xml



PDF is like Machine Learning

- Complex parameter space
- Limited observed data (training)
- Adjust parameters to match observed data
  - *maximize expected performance*

Evaluation of Learning Systems

- Must take sensitivity to training and evaluation inputs into account
  - PDF specializes code according to training data
  - Changing inputs can greatly alter performance
- Performance results must have statistical significance measures
  - Differentiate between gains/losses and noise

Overfitting

- Specializing for the training data too closely
- Exploiting particular properties of the training data that do not generalize
- Causes:
  - insufficient quantity of training data
  - insufficient variation among training data
  - deficient learning system

Overfitting

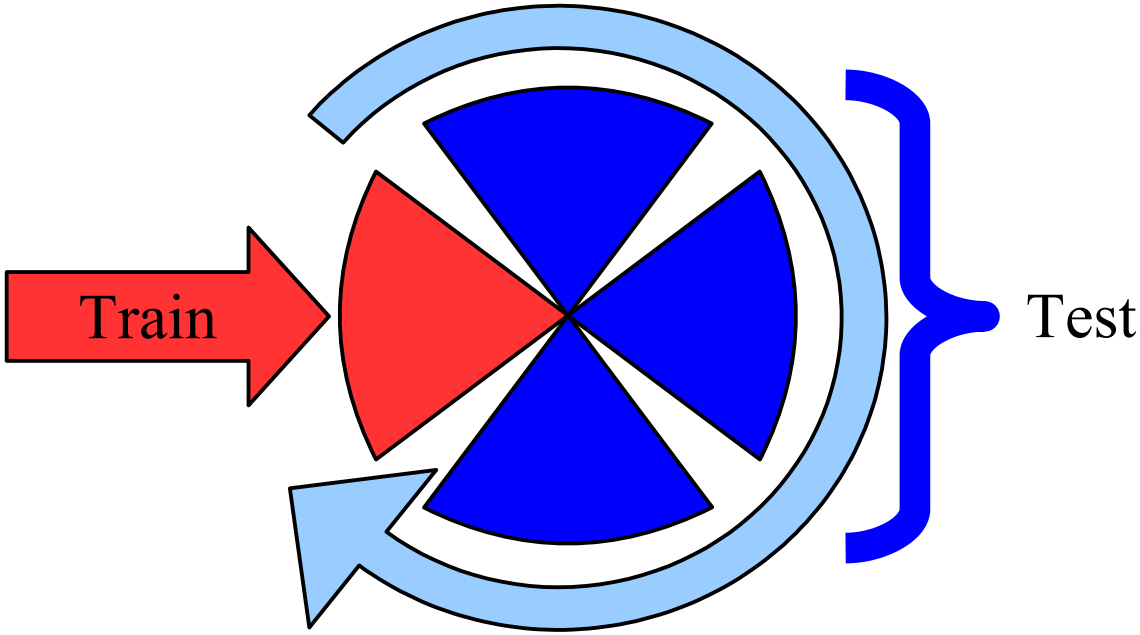
- Currently:
  - ✗ Engineer the compiler to not overfit the single training data (underfitting)
  - ✗ No clear rules for input selection
  - ✗ Some benchmark authors replicate data between train and ref
    - Overfitting can be rewarded!

Criteria for Evaluation

- Predict expected future performance
- Measure performance variance
- Do not reward overfitting
- Same evaluation criteria as ML
  - Cross-validation addresses these criteria

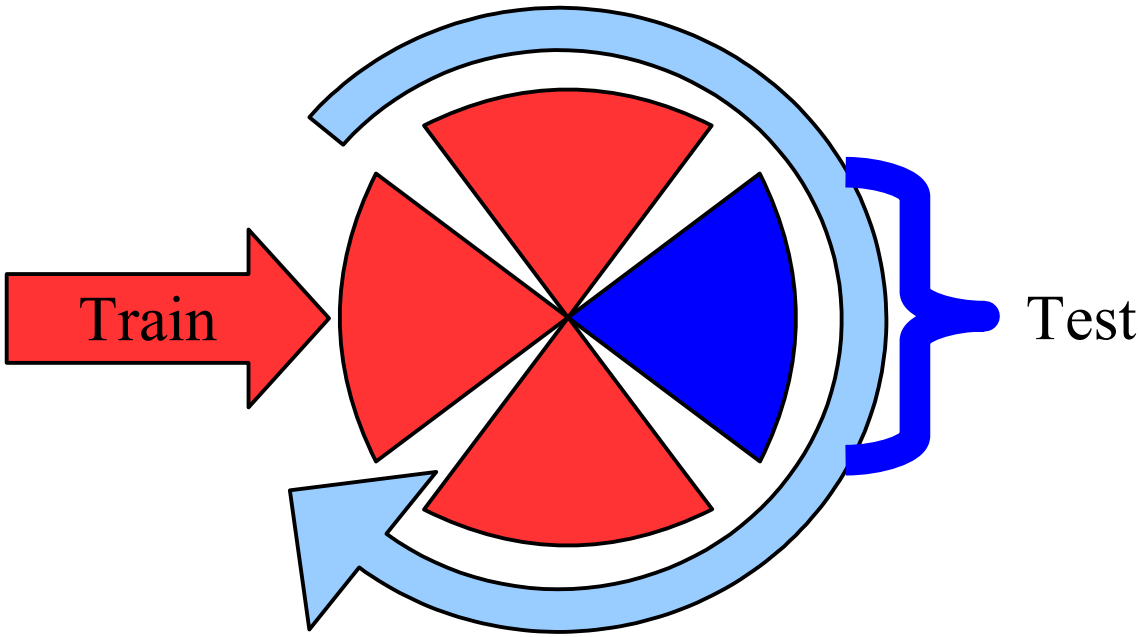
Cross-Validation

- Split a collection of inputs into two or more non-overlapping sets
- Train on one set, test on the other set(s)
- Repeat, using a different set for training



Leave-one-out Cross-Validation

- If little data, reduce test set to 1 input
  - Leave N out: only N inputs in test



Cross-Validation

- The same data is NEVER in both the training and the testing set
  - Overfitting will not enhance performance
- Multiple evaluations allows statistical measure to be calculated on the results
  - Standard deviation, confidence intervals...
- Set of training inputs allows system to exploit commonalities between inputs

Proposed Methodology

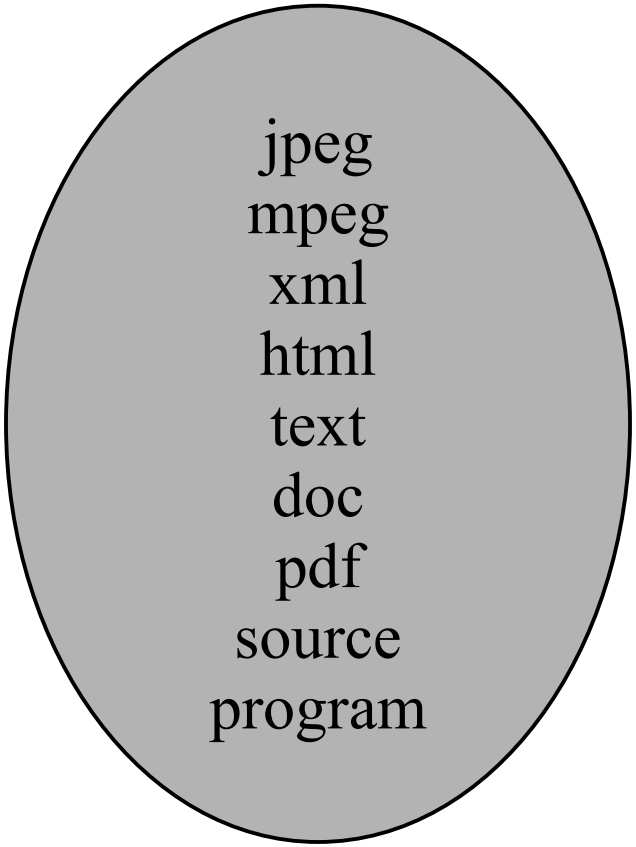
- PDFPeak score, distinct from peak
  - Report with standard deviation
- Provide a PDF workload
  - Inputs used for both training and evaluation, so "medium" sized (~2 min running time)
  - 9 inputs needed for meaningful statistical measures

Proposed Methodology

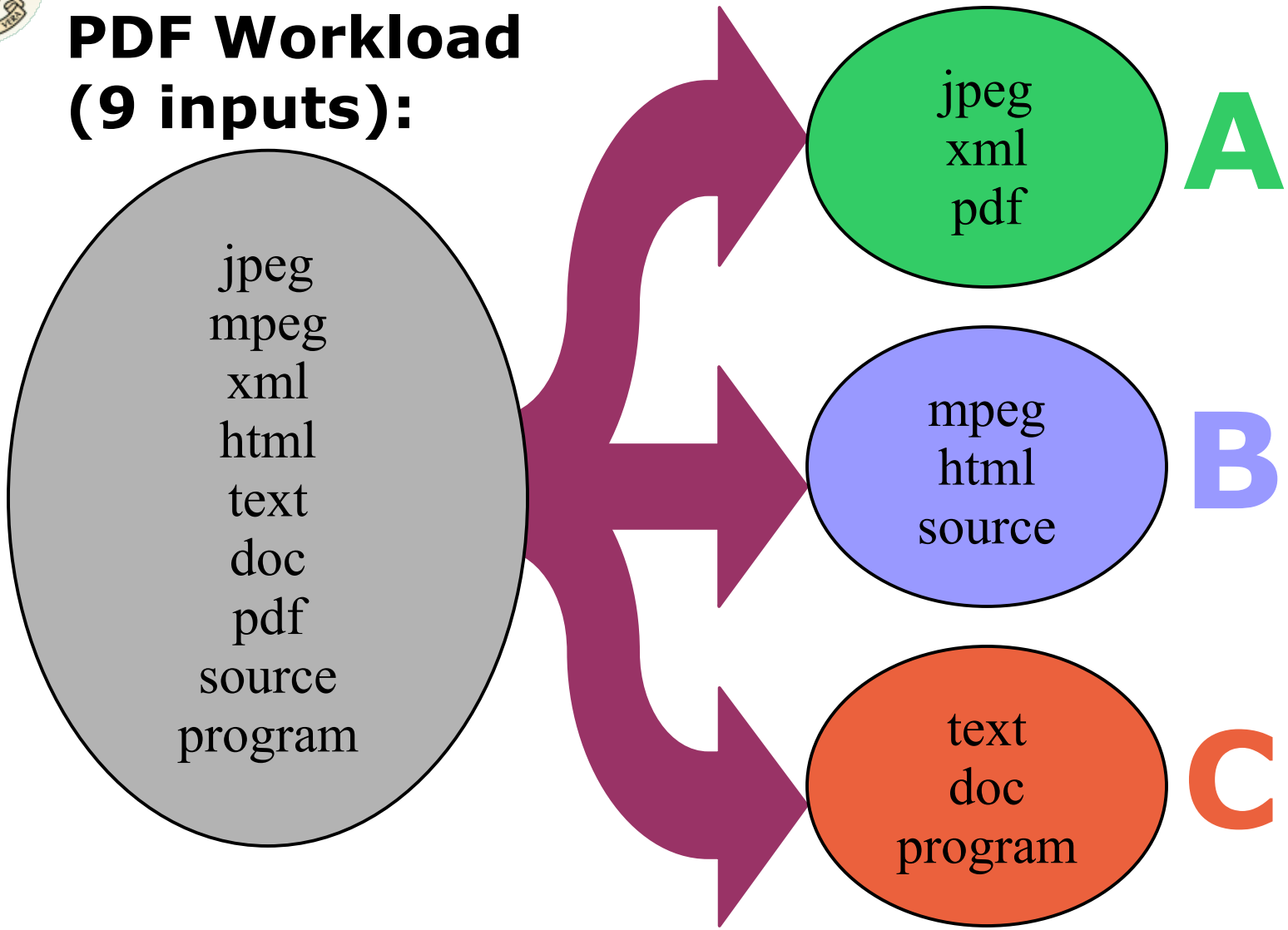
- Split inputs into 3 sets (at design time)
- For each input in each evaluation, calculate speedup compared to (non-PDF) peak
- Calculate (over all evaluations)
  - mean speedup
  - standard deviation of speedups

Example

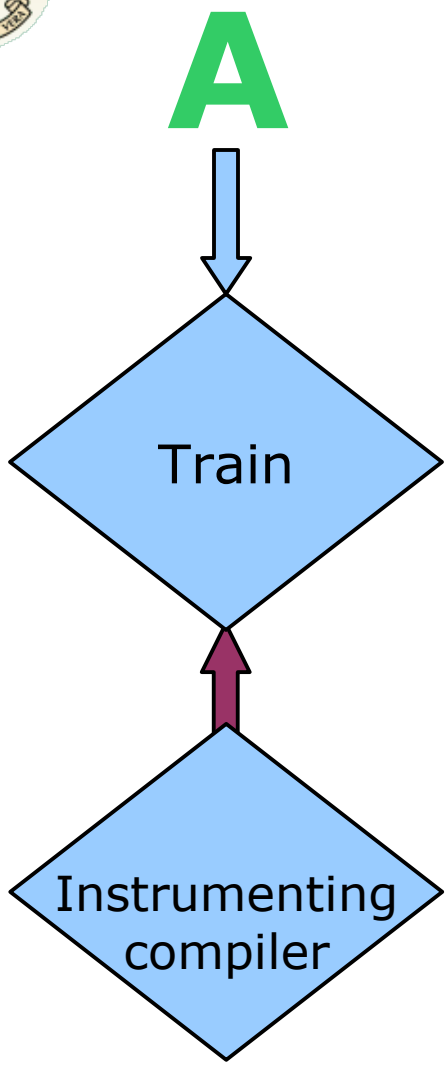
PDF Workload (9 inputs):



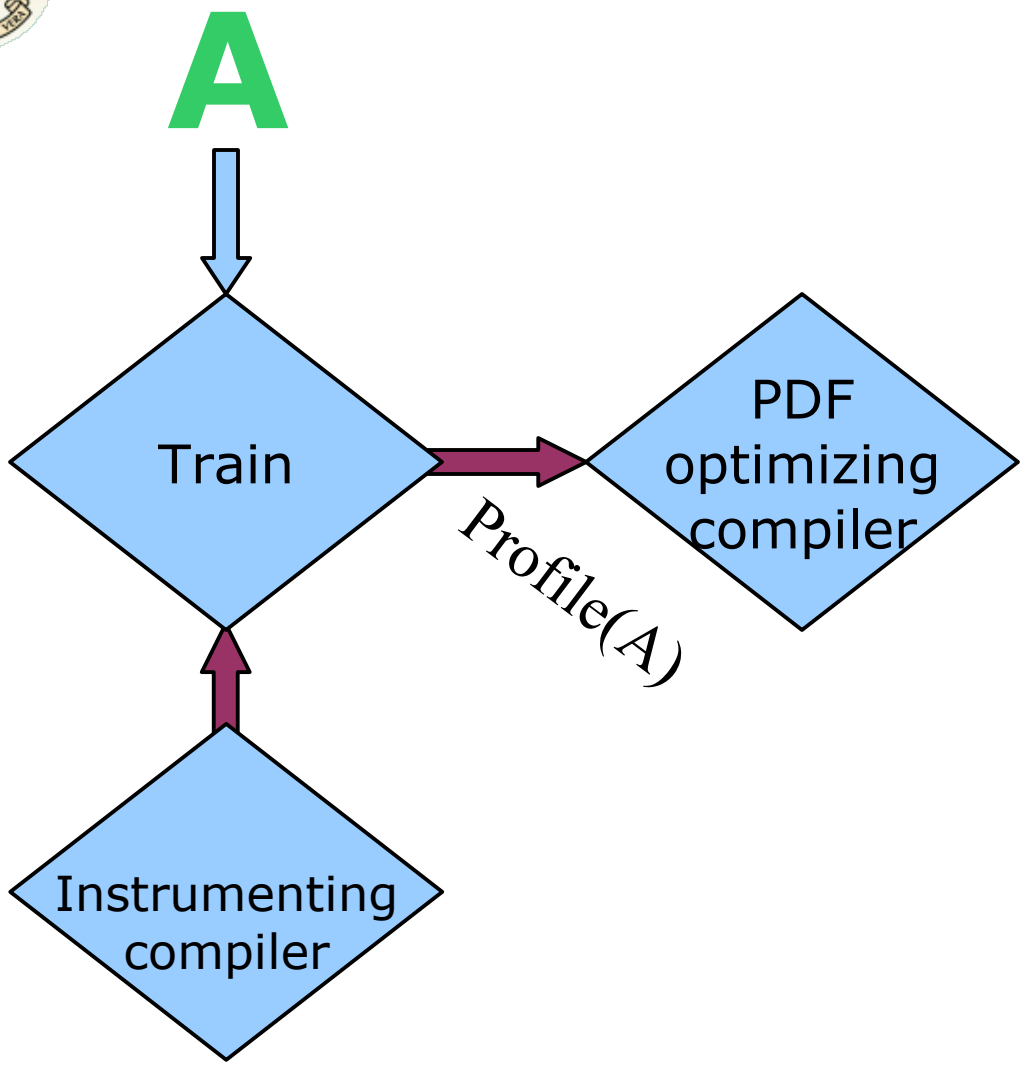
Example – Split workload



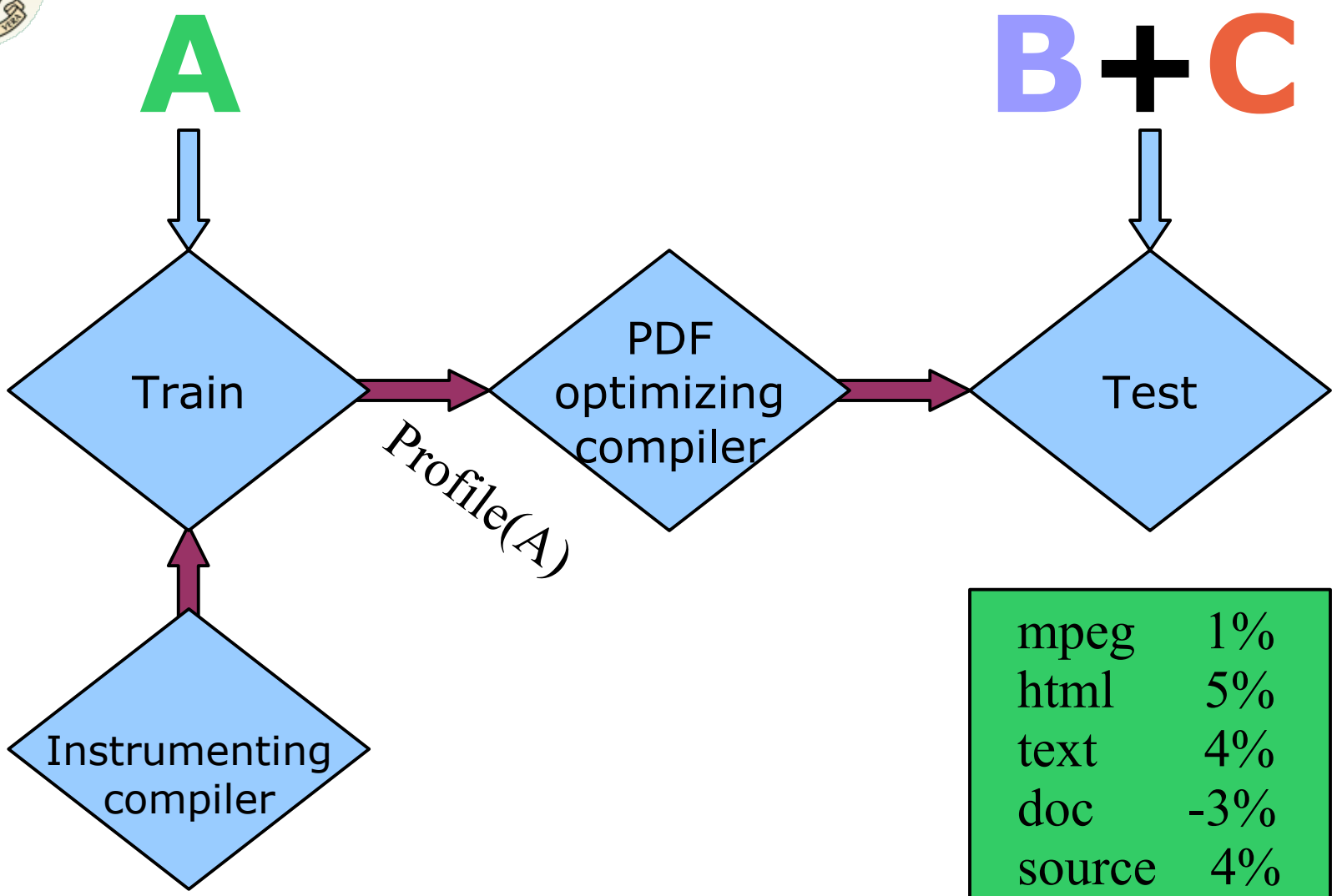
Example – Train and Run



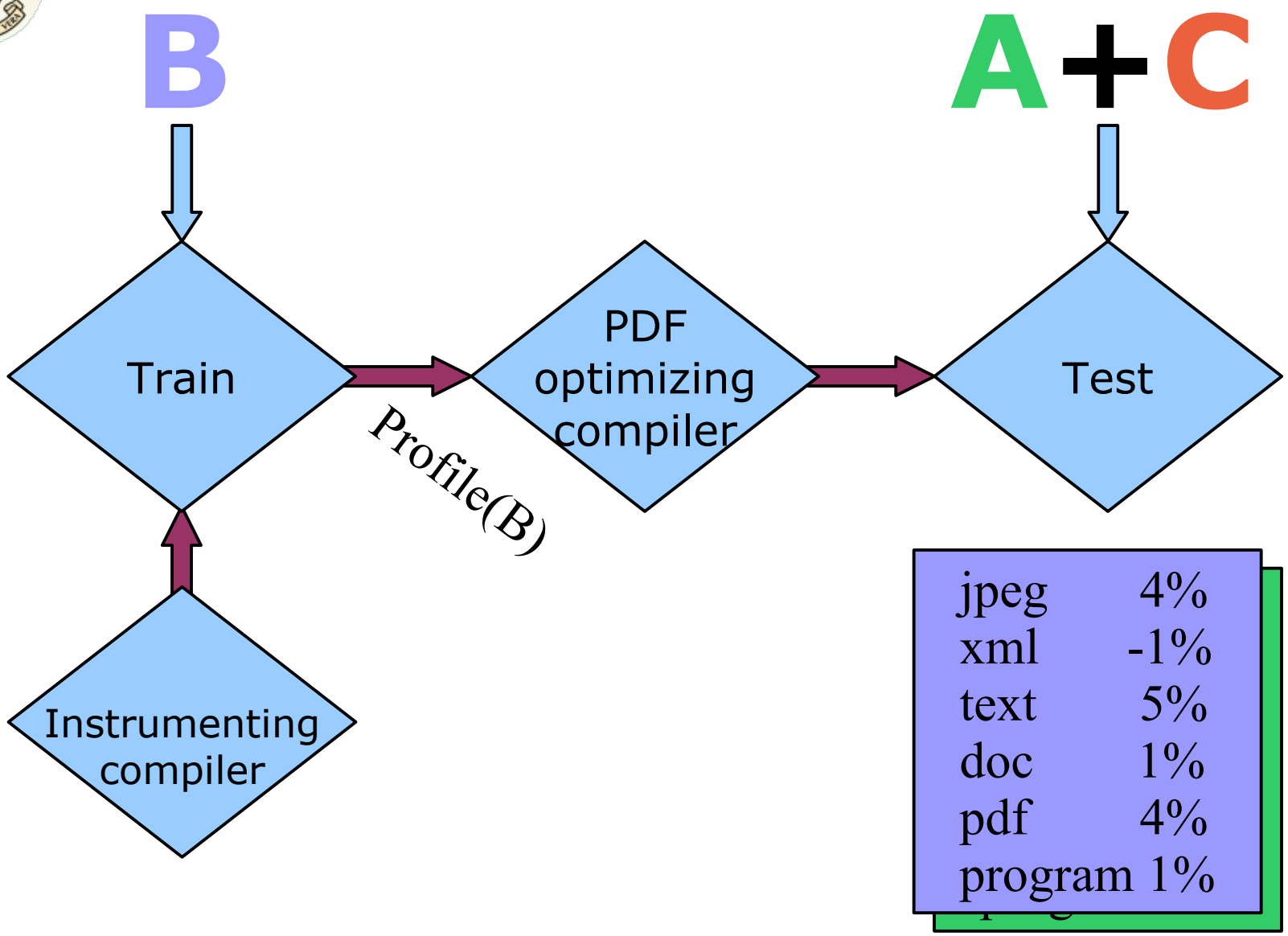
Example – Train and Run



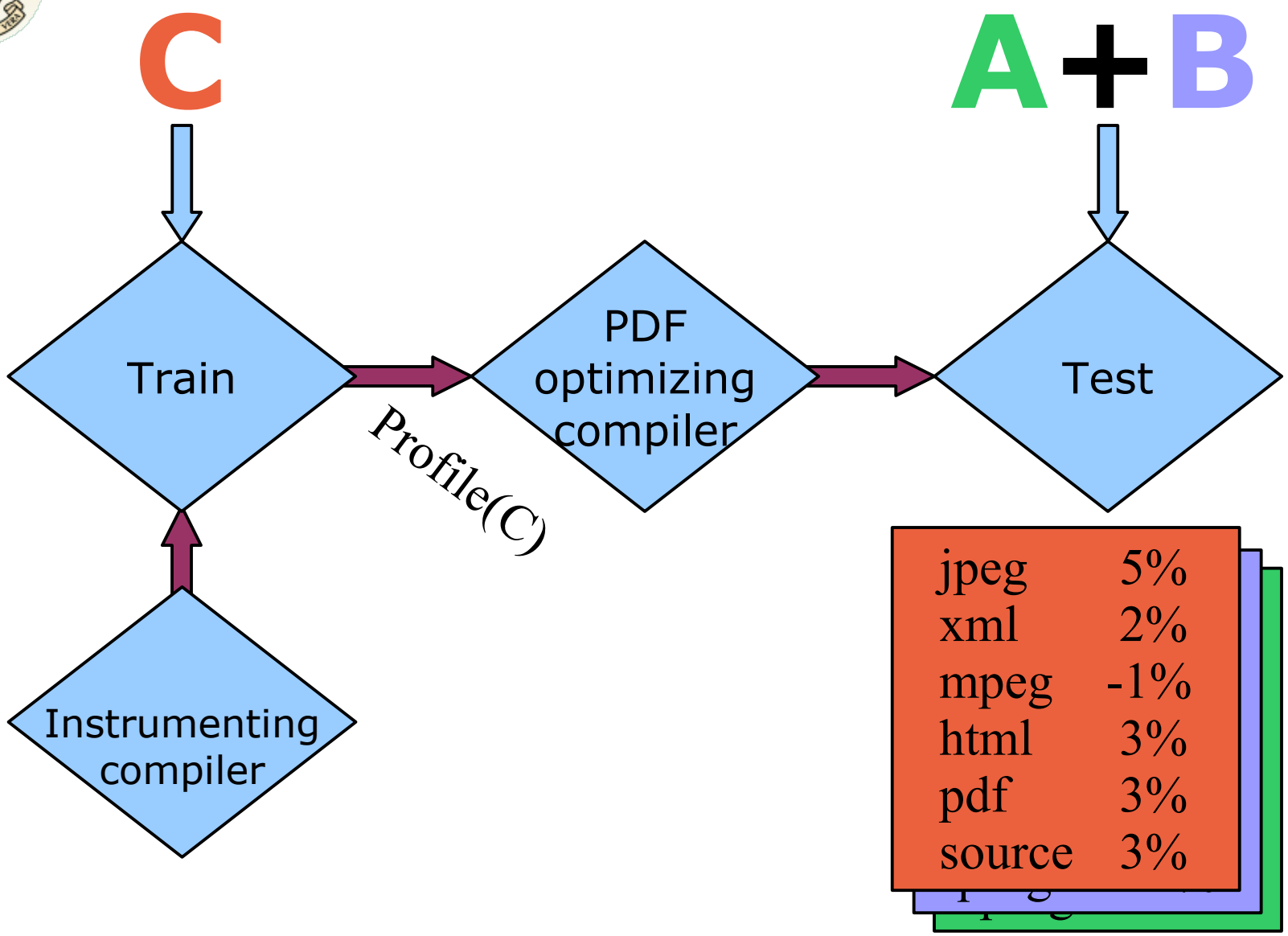
Example – Train and Run



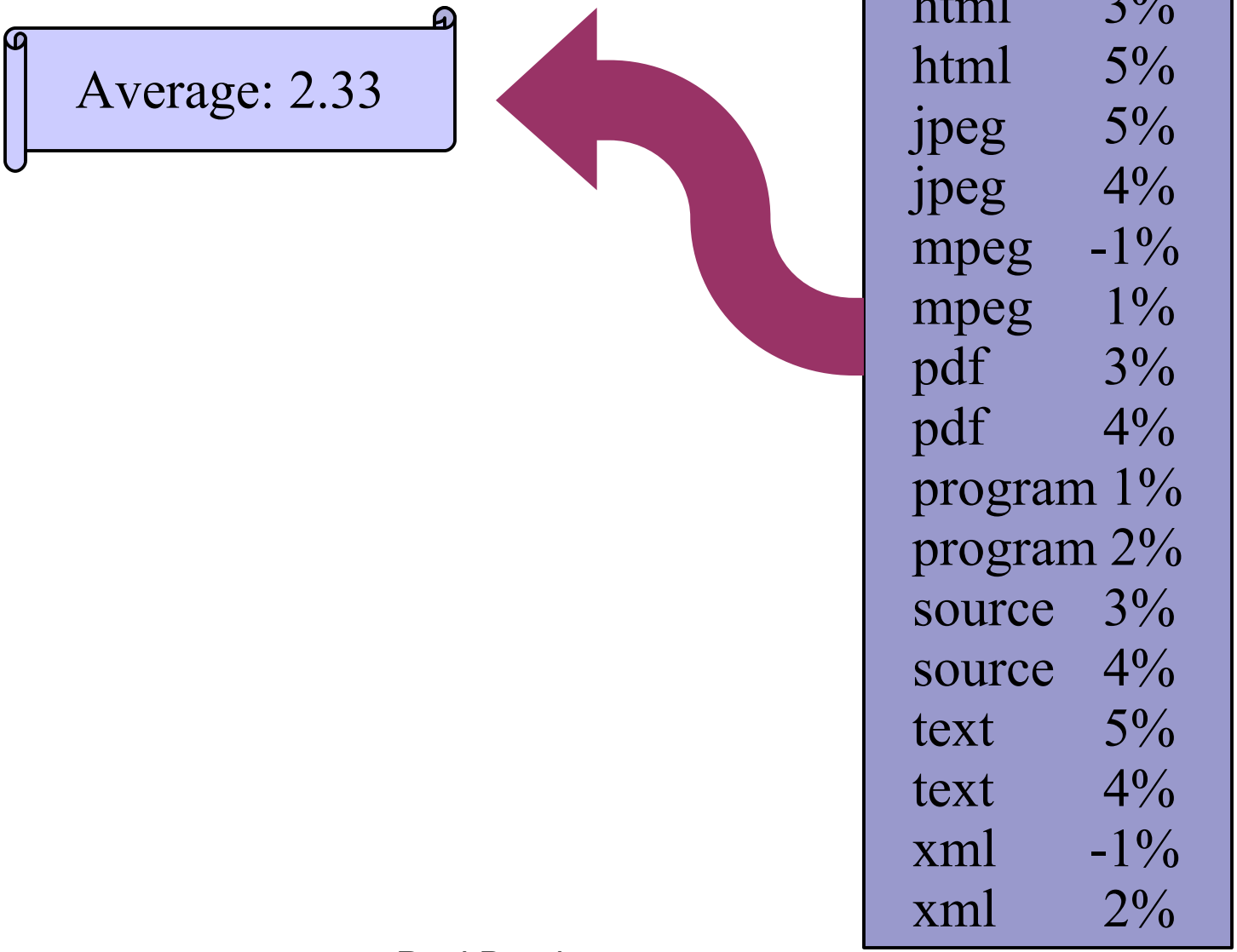
Example – Train and Run



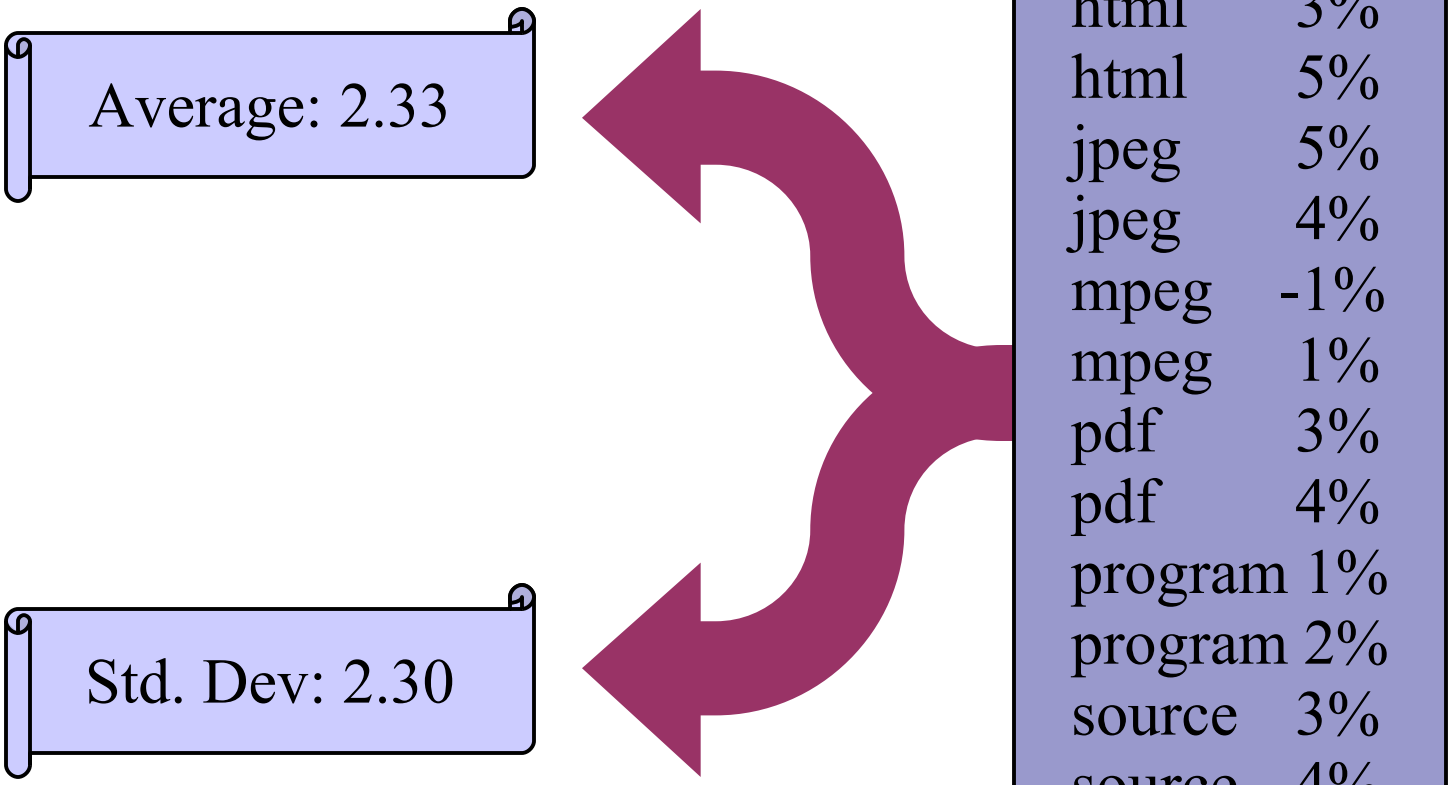
Example – Train and Run



Example – Evaluate



Example – Evaluate



Example – Evaluate

Average: 2.33

PDF improves performance:
- 2.33±2.30%, 17 times out of 25
- 2.33±4.60%, 19 times out of 20

Std. Dev: 2.30

doc	1%
doc	-3%
html	3%
html	5%
jpeg	5%
jpeg	4%
mpeg	-1%
mpeg	1%
pdf	3%
pdf	4%
program	1%
program	2%
source	3%
source	4%
text	5%
text	4%
xml	-1%
xml	2%

Example – Evaluate

PDF improves performance:
- 2.33±2.30%, 17 times out of 25
- 2.33±4.60%, 19 times out of 20

```
(peak_pdf > peak_static)?
  (new_pdf > other_pdf)?
```

Depends on mean and variance of both!

Pieces of Effective Evaluation

- Workload of inputs
- Education about input selection
  - Rules and guidelines for authors
- Adoption of a new methodology for PDF evaluation

Practical Concerns

- Benchmark user
  - Many additional runs, but on smaller inputs
  - Two additional program compilation
- Benchmark author
  - Most INT benchmarks use multiple data, and/or additional data is easily available
  - PDF input set could be used for REF

Conclusion

- PDF is here: important for compilers and architecture, in research and in practice
- The current methodology for PDF evaluation is not reliable
- Proposed a methodology for meaningful evaluation

Thanks

Questions?