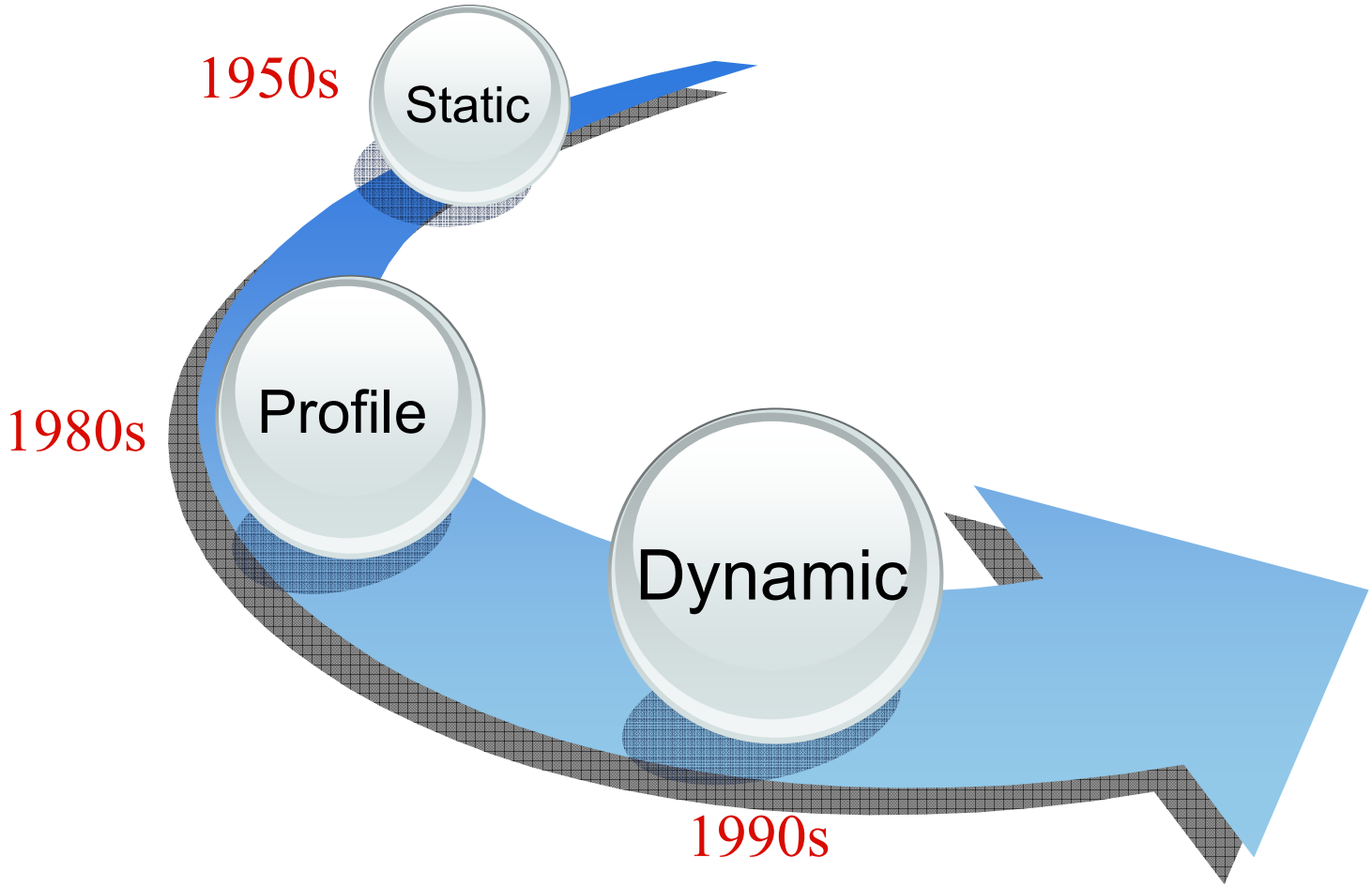
An Input-Centric Paradigm for Program Dynamic Optimizations

Kai Tian, Yunlian Jiang,

Eddy Zhang, Xipeng Shen

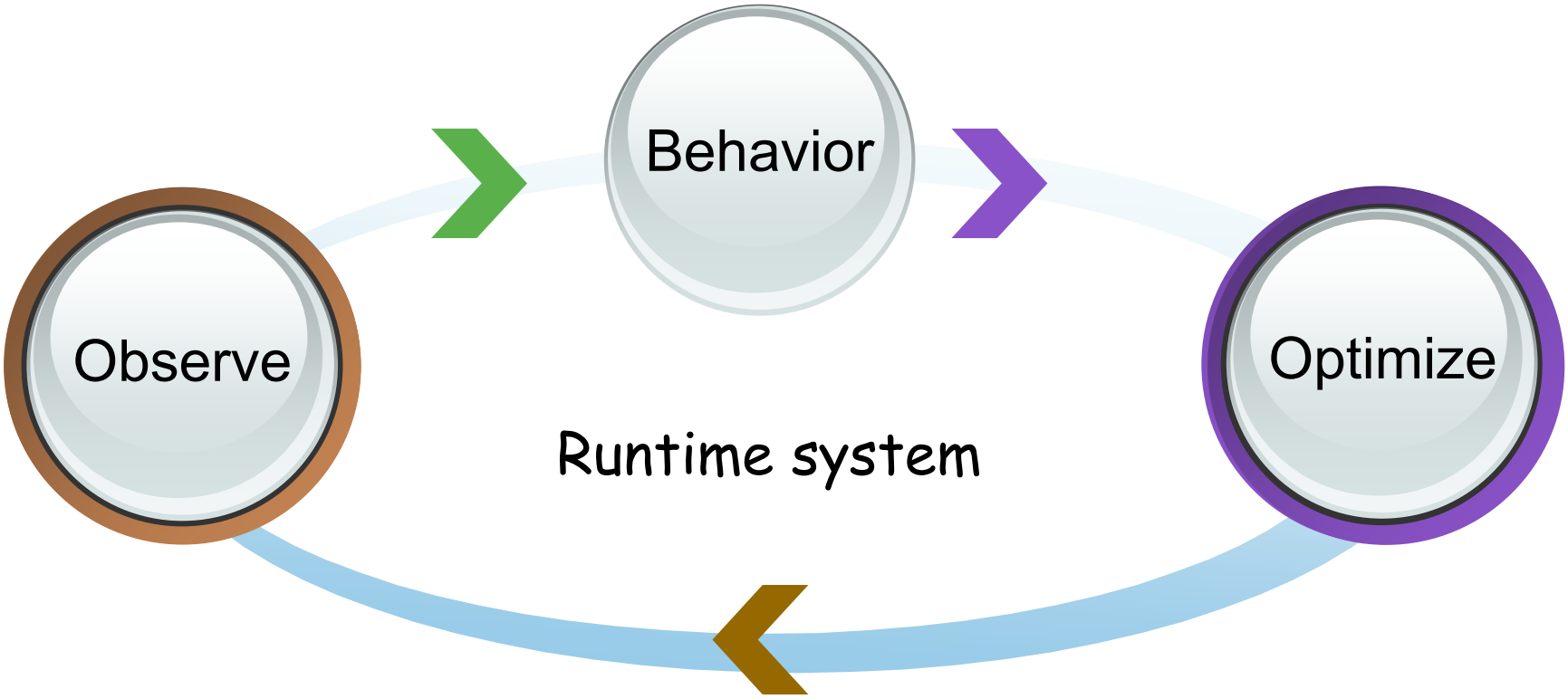
College of William and Mary

Program Optimizations



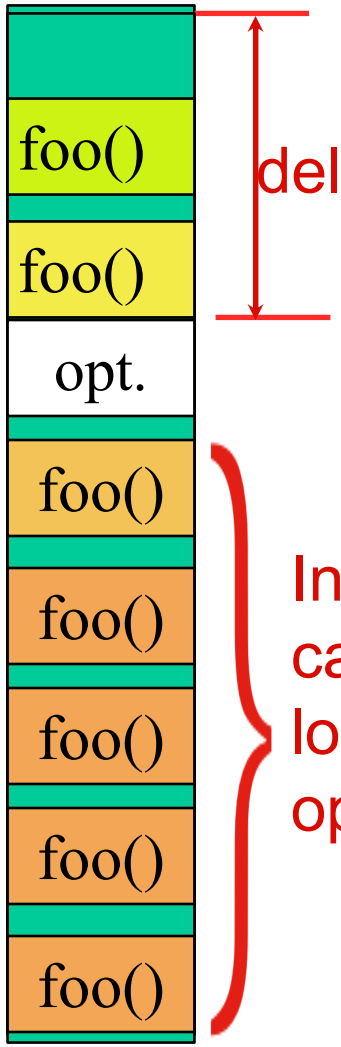
Dynamic Optimizations

Widely used in Java, C#, etc.



Drawback of Dynamic Opt

```
while (...){
   foo ();
}
```



delay

Runtime overhead

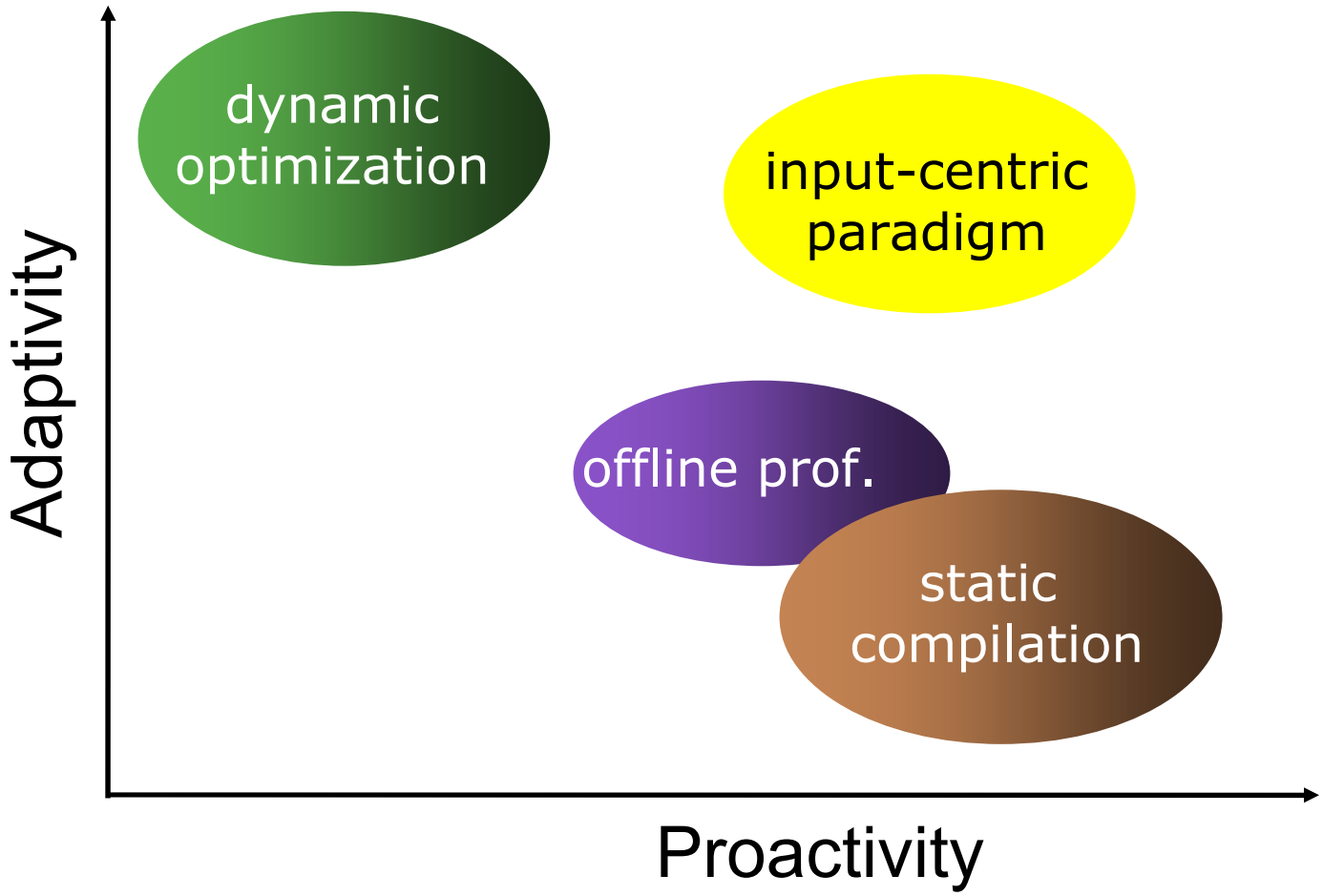
Reactive scheme

Inferior performance caused by local view-based optimizations

47% on J9 [Arnold+' 05]

21% on JikesRVM [Mao+' 09]

Adaptivity-Proactivity Dilemma



Outline

- Why input-centric?
- Input-centric paradigm
- Evaluation
- Related work
- Conclusion

Prerequisite for Optimizations

Accurate prediction of how programs would behave.

Program Behaviors

(method calling freq, locality,
loop trip counts...)

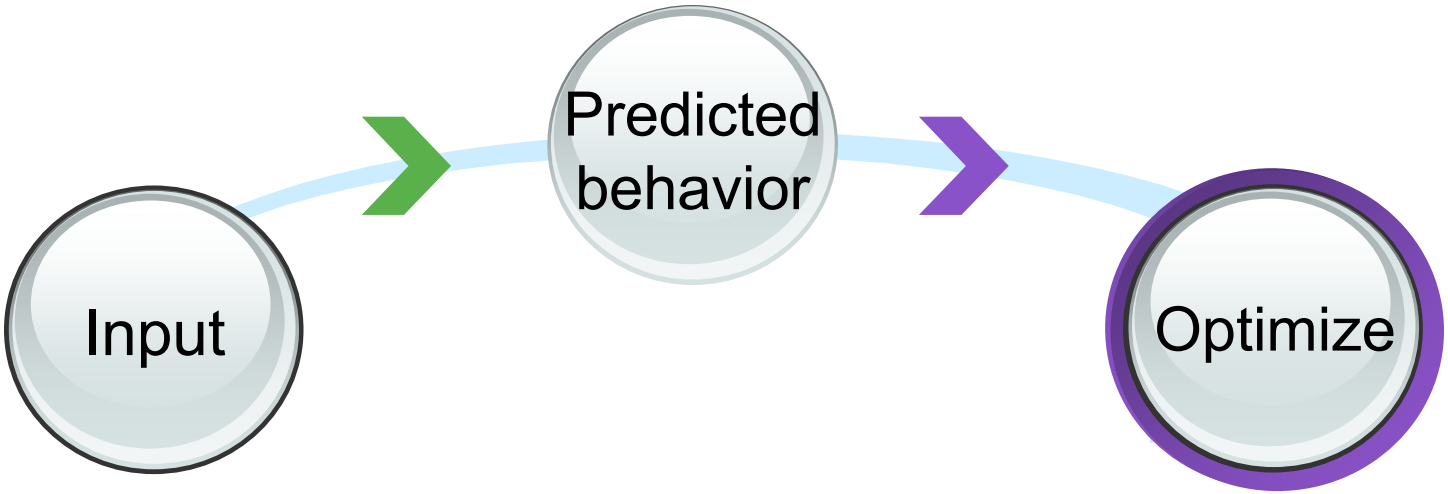
What Decide Program Behavior?

Prog Beh = Code + Inputs + Running Environments

only deciding factor given a program on a machine

- Command-line arguments
- Interactively input data
- Input files
- ...

Input-Centric Opt Paradigm



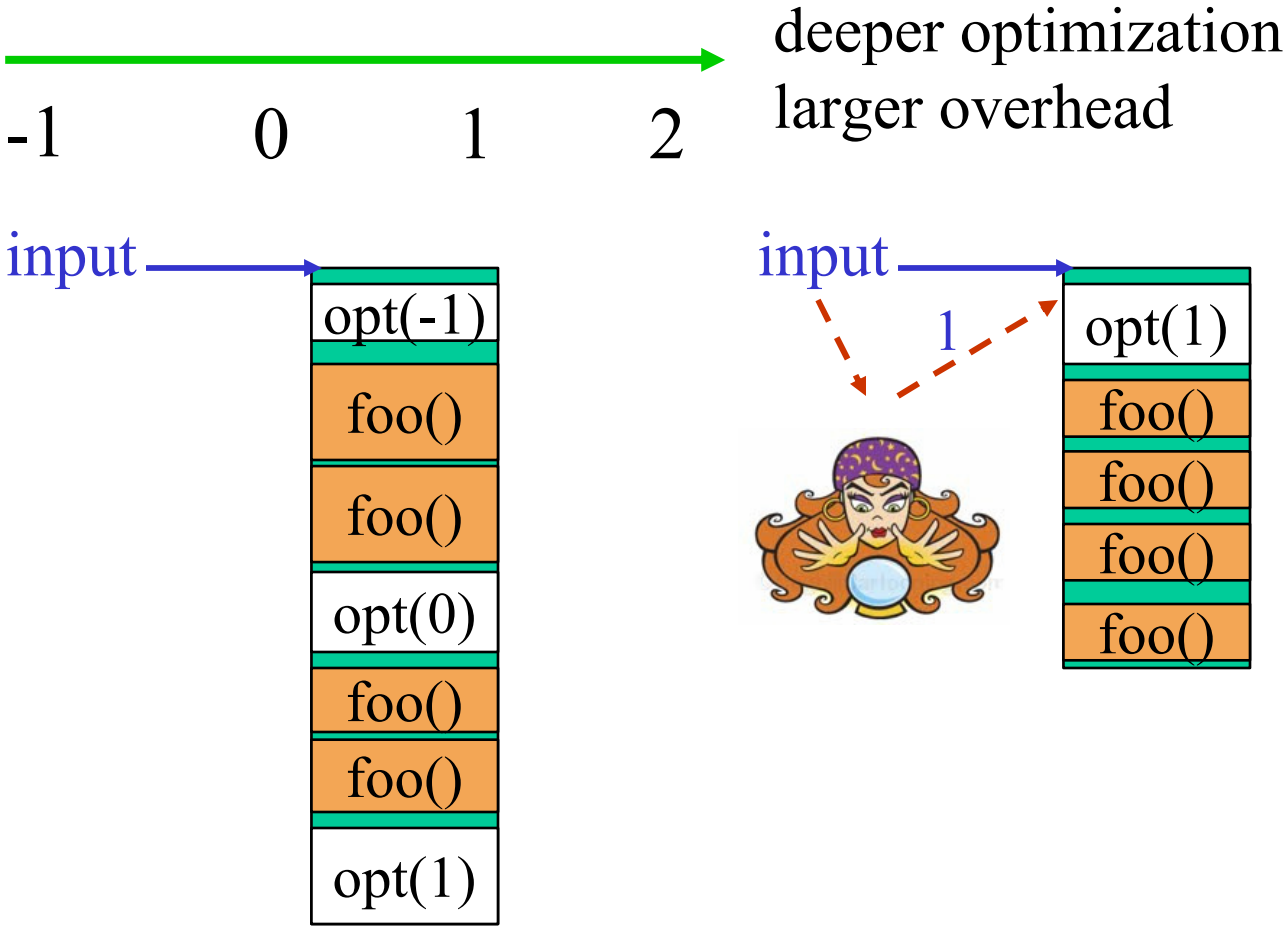
Idea: Use program inputs to trigger runtime behavior prediction and proactive optimizations

Proactivity: Early optimize based on prediction

Adaptivity: Input-specific optimization

Benefits for JIT

- JIT in JikesRVM

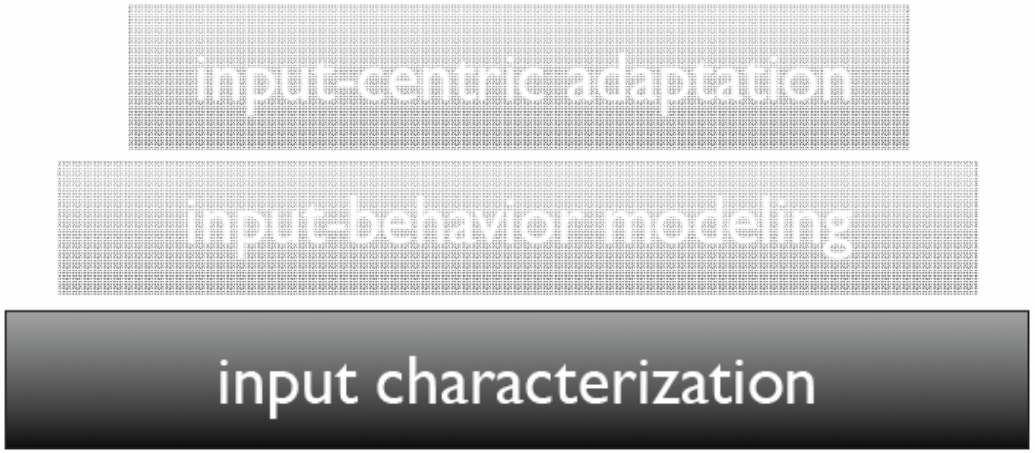


** Thanks to coghillcartoning.com for the image.*

Challenges

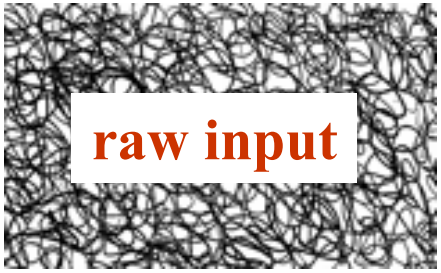
- Complexities in inputs
- Complexities in relations
- Integration in runtime

Techniques to Realize Input-Centric Paradigm



Input Characterization

- Goal



input feature vector
< feature 1, feature 2, ..., feature k >

- Solution
  - Seminal Behaviors [Jiang+: CGO'10]
    - Exploit strong correlations among program behaviors

```
main(int argc, char * argv){
  ...
  mesh_init (dataFile,mesh,refMesh);
  genMesh (mesh,0,mesh->vN);
  verify (mesh, refMesh);
}

// recursive mesh generation
void genMesh (Mesh *m, int left, int right){
  if (right>3+left){
    genMesh (m, left, (left+right)/2);
    genMesh (m, (left+right)/2+1, right);
    ...}
  ...
}

void verify (Mesh *m, Mesh *mRef){
  ...
  for (i=0, j=0; i< m->edgesN; i++){
    ...
  }
}
```

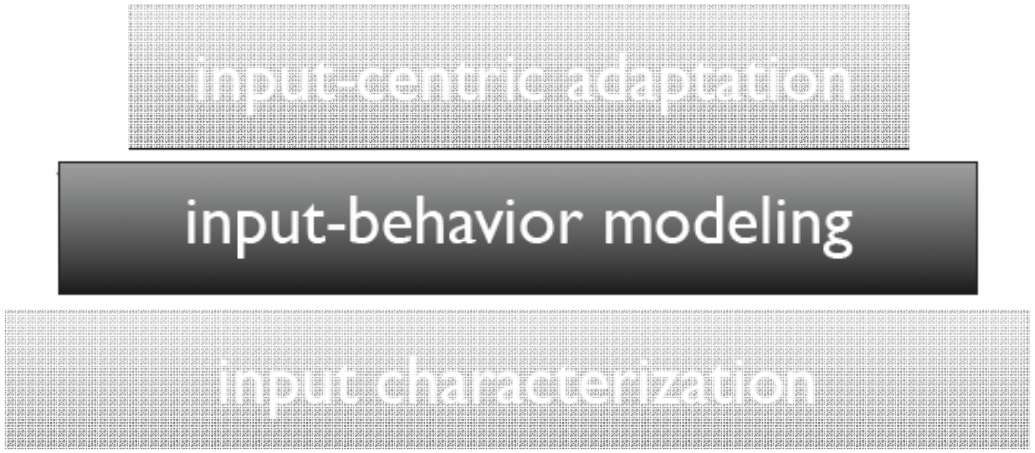
```
Mesh * mesh_init
(char * initInfoF, Mesh* mesh, Mesh* refMesh)
{
    // open vertices file, read # of vertices
    FILE * fdata = fopen (initInfoF, "r");
    fscanf (fdata, "%d, %\n", &vN);
    mesh->vN = vN;
    v = (vertex*) malloc (vN * sizeof(vertex));
    // read vertices positions
                                    v[i].x, &v[i].y);
    ...}
    // sort vertices by x and y values
    for (i=1; i< vN; i++){
        for (j=vN-1; j>=i; j--){
          ...}
    }
    while (!feof(fd)){
      ...
     // read edges into refMesh for
    // later verification
    }
}
```

Seminal Behaviors

Seminal Behaviors Identification

- Through statistical learning
- Fully automatic framework
- Details in [Jiang+:CGO'10].

Techniques to Realize Input-Centric Paradigm



Input Behavior Modeling

- Problem formulation
  - To construct predictive models
    - Target Behaviors = f (Seminal Behaviors)
- Solution: Cross-run incremental learning
  - Target is categorical beh. (e.g., opt levels)
    - Classification Trees
  - Target is numerical beh. (e.g., calling freq.)
    - Linear Regression (LMS)
    - Regression Trees

Special Challenges

- Categorical vs. numerical features
  - Data types
  - Number of unique values in training data sets
- Feature selection
  - Classification & regression trees
    - Filter out unimportant features automatically
  - LMS regression
    - PCA (when all features numerical)
      - Select directions showing large variations
    - Stepwise selection (otherwise)
      - Continuously add features that improve prediction

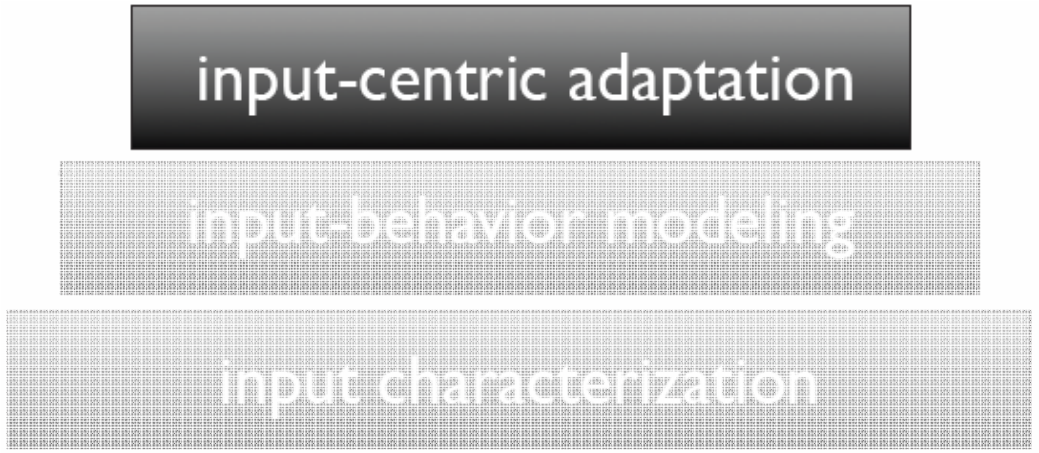
Risk Control

- Prevent effects of wrong predictions
  - Fine-grained discriminative prediction

```
Keep assessing confidence level of each input
subspace;
if (confidence_level > θ)
        Do prediction;
else
        Fall back to default reactive strategy;
```

- Model evolvement
  - Behavior models / confidence values evolve on new runs
  - Model retrained using expanded data set

Techniques to Realize Input-Centric Paradigm



Evaluation 1: JikesRVM opt

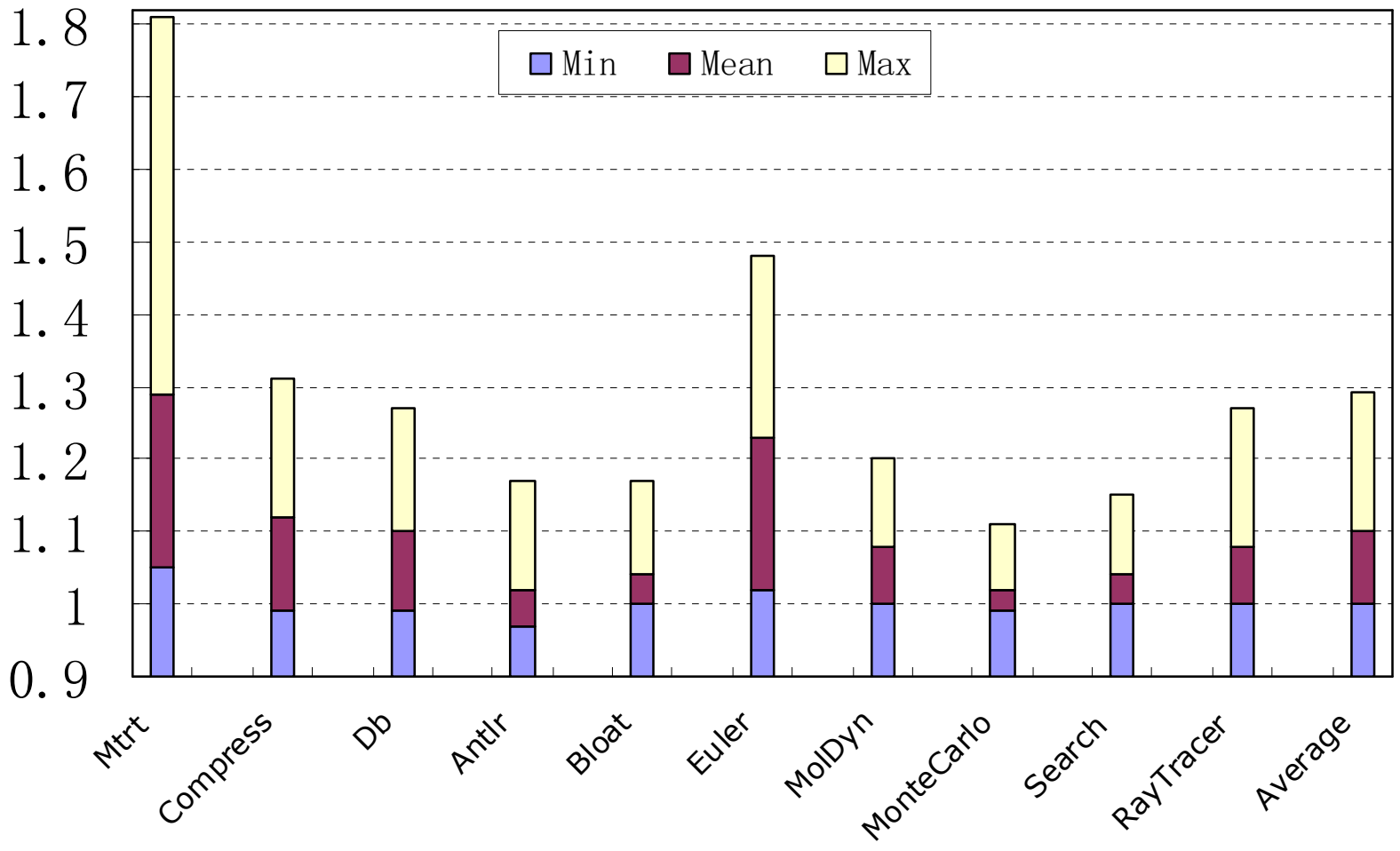
- Machine
  - Intel Xeon E5310, Linux 2.6.22
- Java Runtime
  - Modified JikesRVM 3.1.0
- Benchmarks
  - 10 Java programs from Dacapo, Grande, JVM98
- Inputs
  - Extra inputs from [Mao+:CGO'09]

Prediction Accuracy for Java

Program	# of inputs	# of sem.beh.	Prediction accuracy		
			opt level	call freq	min heap
Compress	20	2	0.99	0.93	0.99
Db	54	4	0.84	0.98	0.96
Mtrt	100	2	0.97	0.89	0.84
Antlr	175	39	0.95	0.95	0.96
Bloat	100	7	0.96	0.76	0.99
Euler	14	1	0.99	0.99	0.98
MolDyn	15	2	0.98	0.83	0.98
MonteCarlo	14	1	0.99	0.98	0.99
Search	9	2	0.99	0.97	0.99
RayTracer	12	1	0.98	0.9	0.98
Average	**51.3**	**6.1**	**0.96**	**0.92**	**0.97**

Speedup in JikesRVM

Baseline: default JikesRVM



Evaluation 2 : Dynamic Version Selection

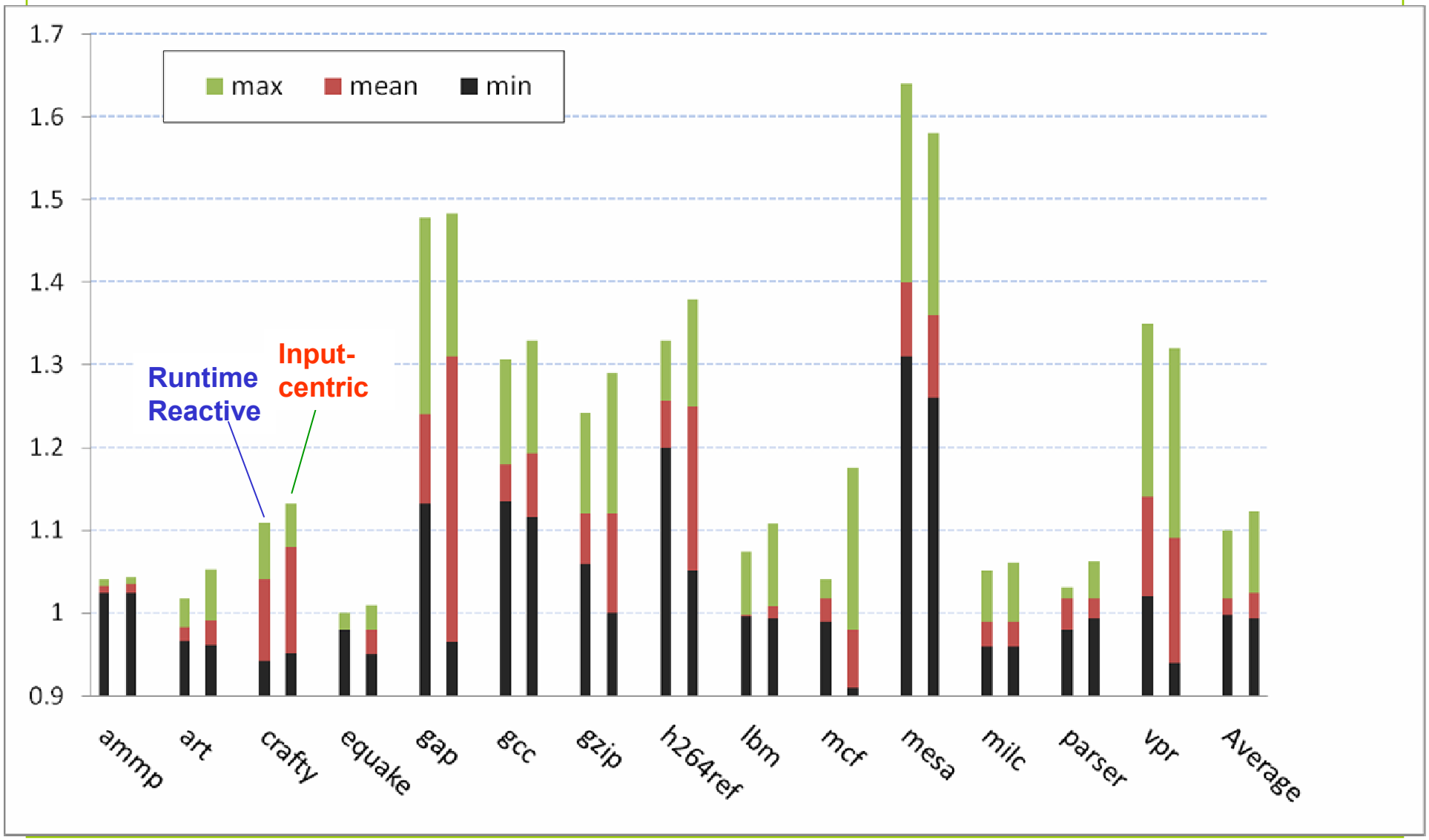
- Input-centric adaptation
  - Model from input features to suitable versions
  - Predict best version to run during runtime
- Reactive approach [Chuang+:07]
  - Timing each version to select the best during runtime

Experiment Setting

- Versions creation
  - IBM XL C compiler
  - 5 code versions from feedback-driven opt
- Machines
  - IBM Power5
  - AIX 5.3

Speedup in Version Selection

Baseline: static compile with highest opt level



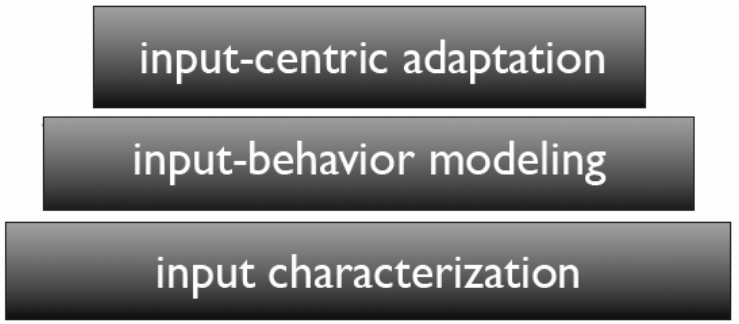
Discussions

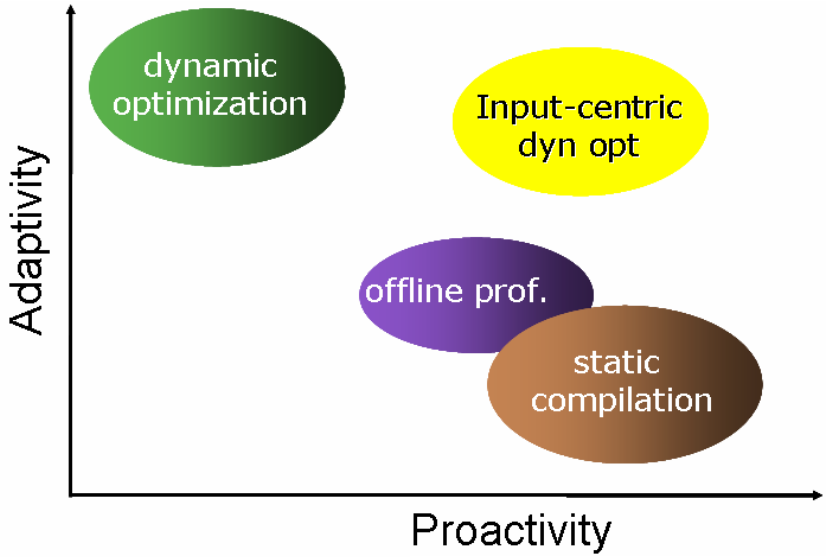
- Three steps for input-centric optimizations
  - Profile collection (offline)
  - Seminal beh recog. & input-beh model construction (offline)
  - Proactive behavior prediction & optimizations (online)
- Input-centric paradigm is fundamental
  - May benefit many other optimizations
    - Anywhere runtime adaptation is needed
- Not conflict with phase changes
- Complement to reactive dynamic optimizations

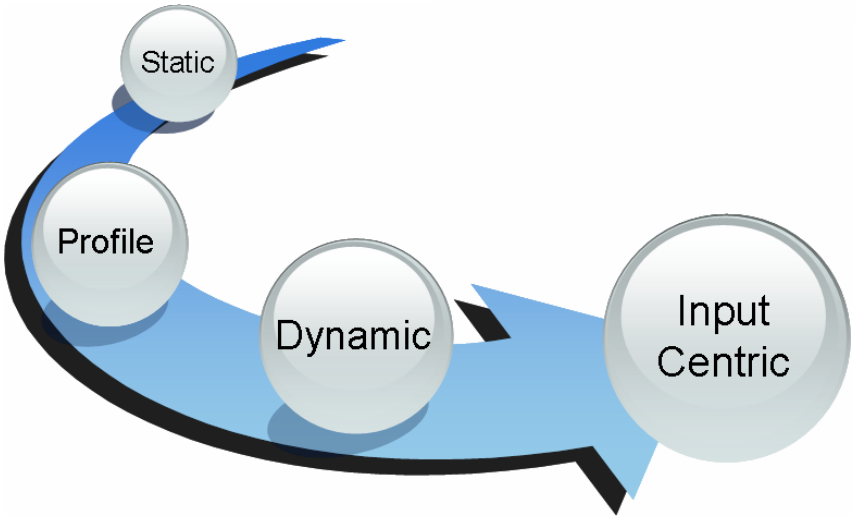
Related Work

- Library development
  - ATLAS [Whaley+:01], Sorting [Li+:CGO04], FFTW [Frigo+: IEEE'05], SPIRAL [M. Puschel+: IEEE'05], STAPL [Thomas+: PPOPP'05]
- General-purpose programming
  - Seminal behavior exploration [Jiang+: CGO'10]
  - Specification language (XICL) to capture input features [Mao+:CGO'09]
- Connections among program behaviors
  - Between loops and method hotness [Namjoshi+: VEE'10]

Conclusions







Thanks!

Questions?

Kai Tian
ktian@cs.wm.edu