An Input-Centric Paradigm for Program Dynamic Optimizations

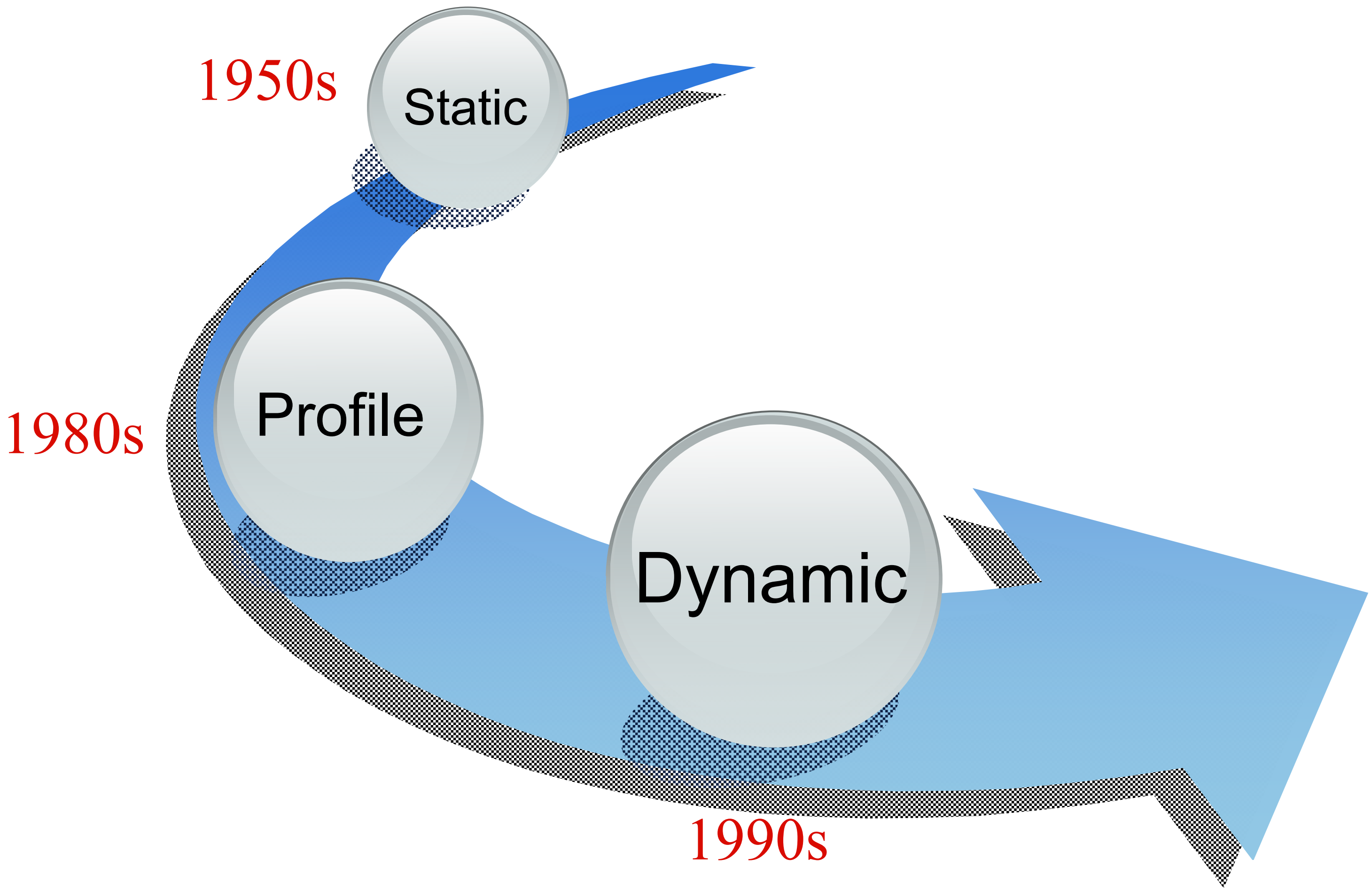
Kai Tian, Yunlian Jiang, Eddy Zhang,

Xipeng Shen (presenter)

College of William and Mary

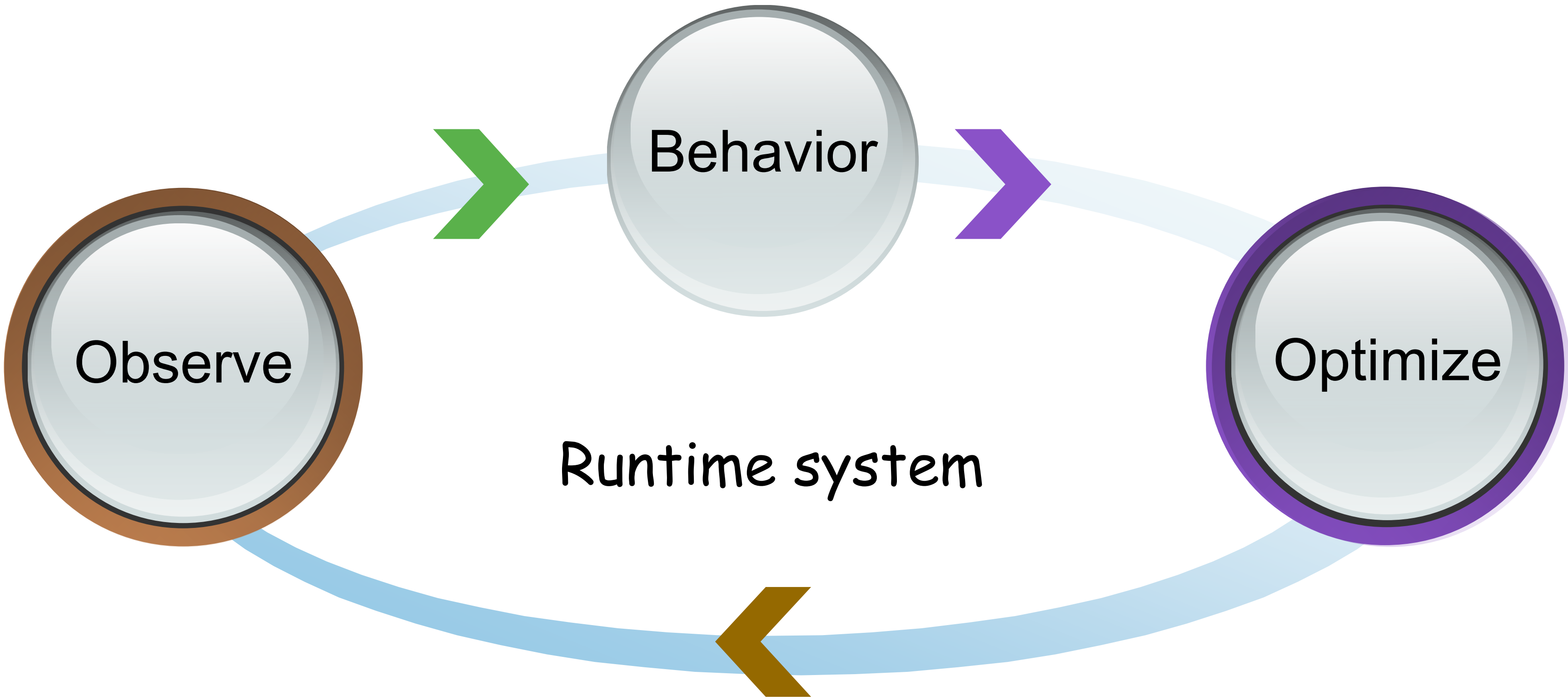
(Published in OOPSLA'10)

Program Optimizations



Dynamic Optimizations

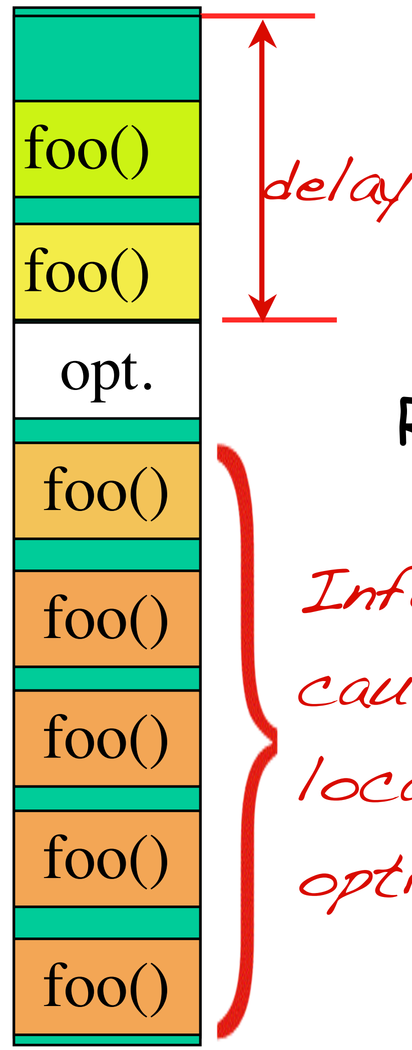
Widely used in Java, C#, etc.



Current Limitations

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

Runtime overhead



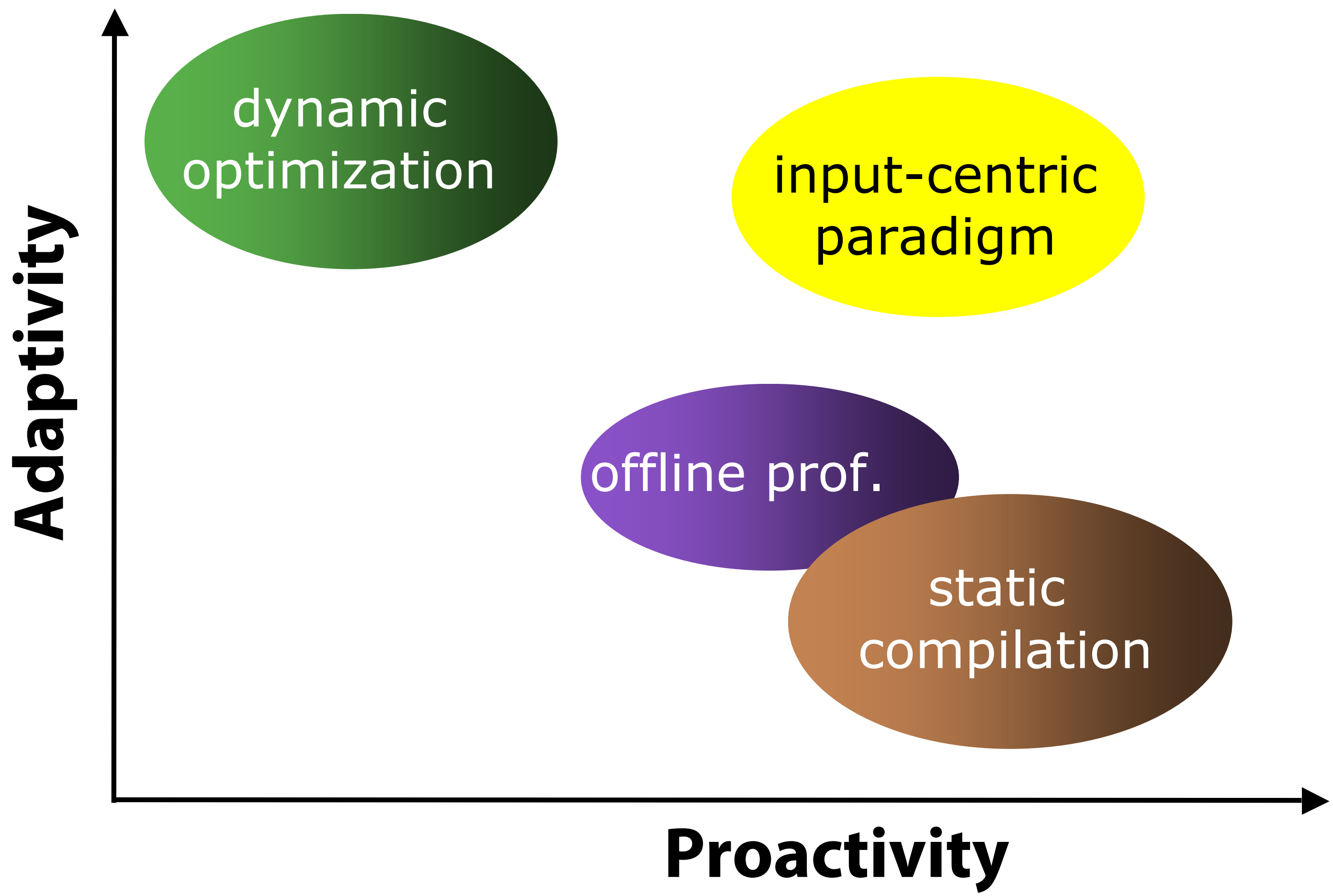
Reactive scheme

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 <u>how programs would behave</u>.

Program Behaviors

↓

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

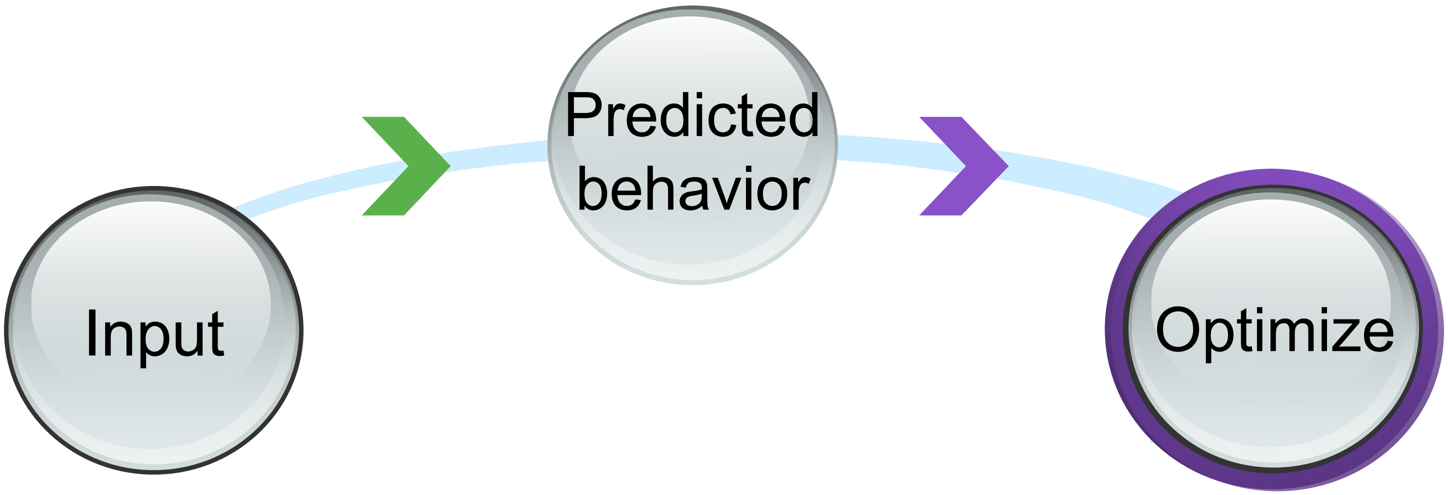
What Decide Program Behavior?

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



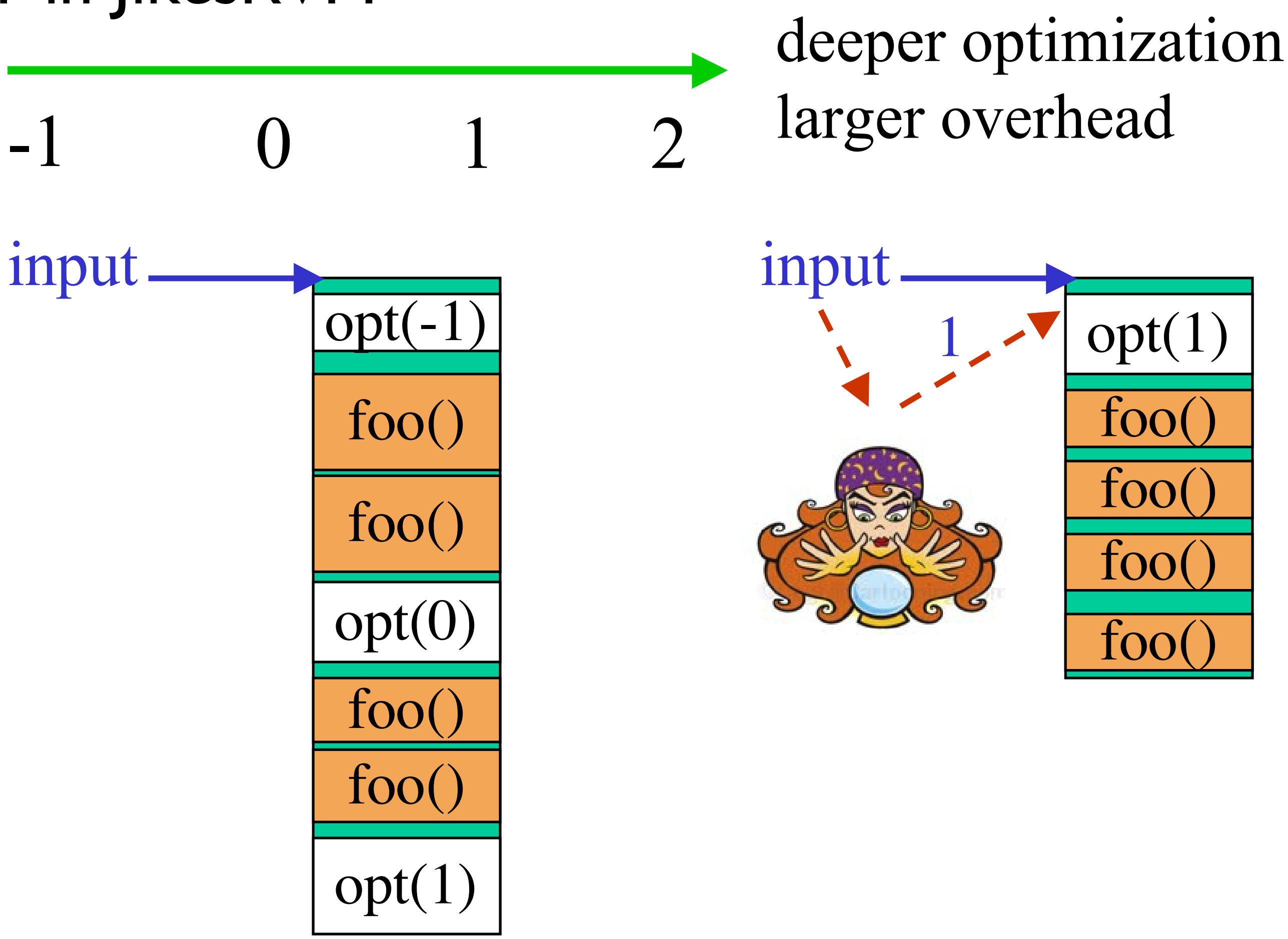
Idea: Use program inputs to trigger runtime behavior prediction and proactive optimizations

Proactivity: Early optimize based on prediction

Adaptivity: Input-specific optimization

Benefits for JIT

- JIT in JikesRVM



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

input

behaviors

Challenges

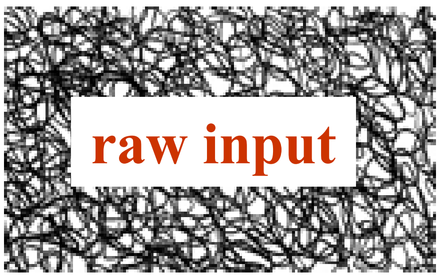
- Complexities in inputs
- Complexities in relations
- Integration in runtime

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

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 machine learning
  - Target beh. is categorical (e.g., opt. levels)
    - Classification Trees
  - Target beh. is numerical (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 > Threshold)
   Do prediction;
else
   Fall back to default reactive strategy;
```

Details in [Tian+:OOPSLA'10].

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]

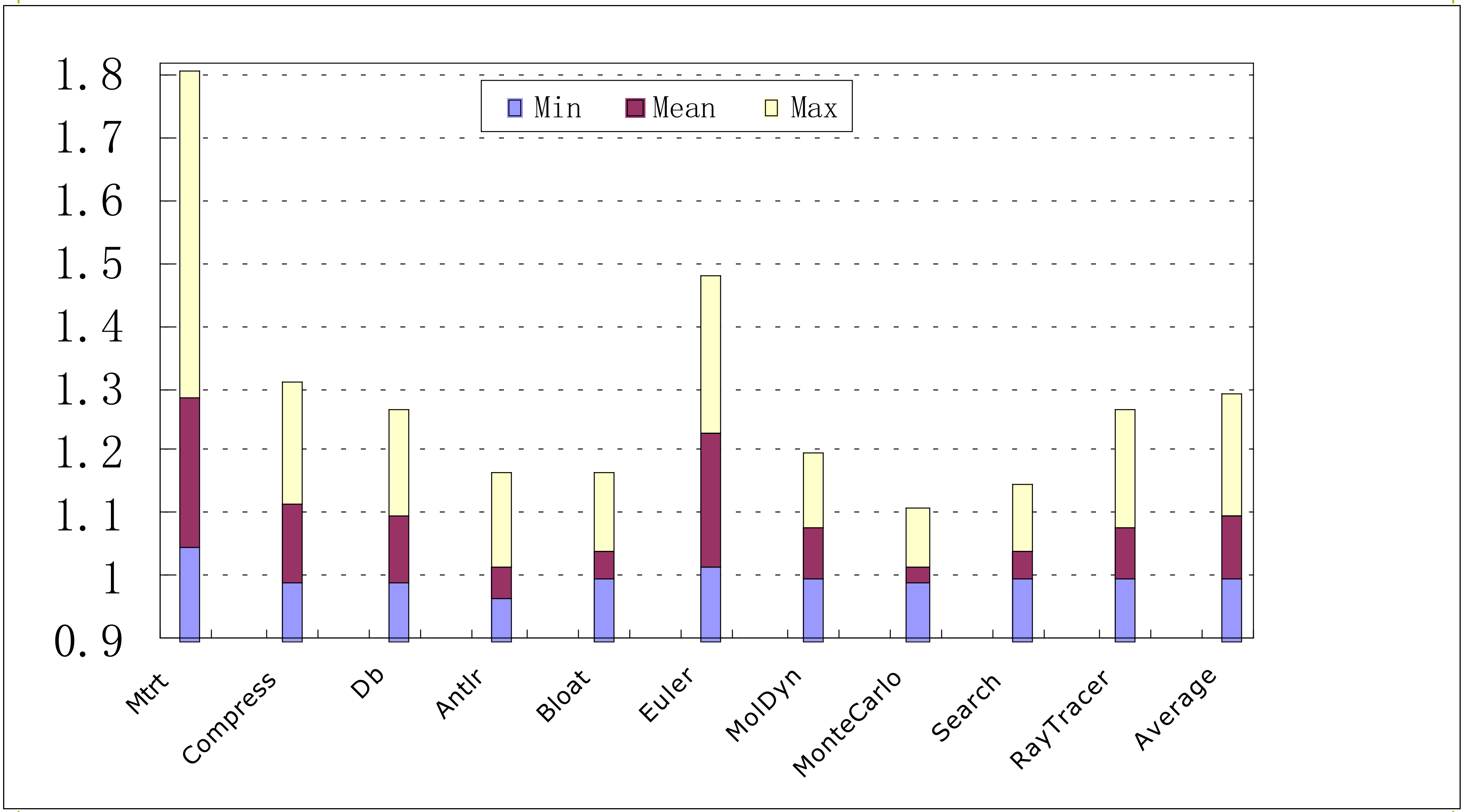
Prediction Accuracy for Java

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**

10-fold cross-validation

Speedup in JikesRVM

Baseline: default JikesRVM



Evaluation 2 : Dynamic Version Selection

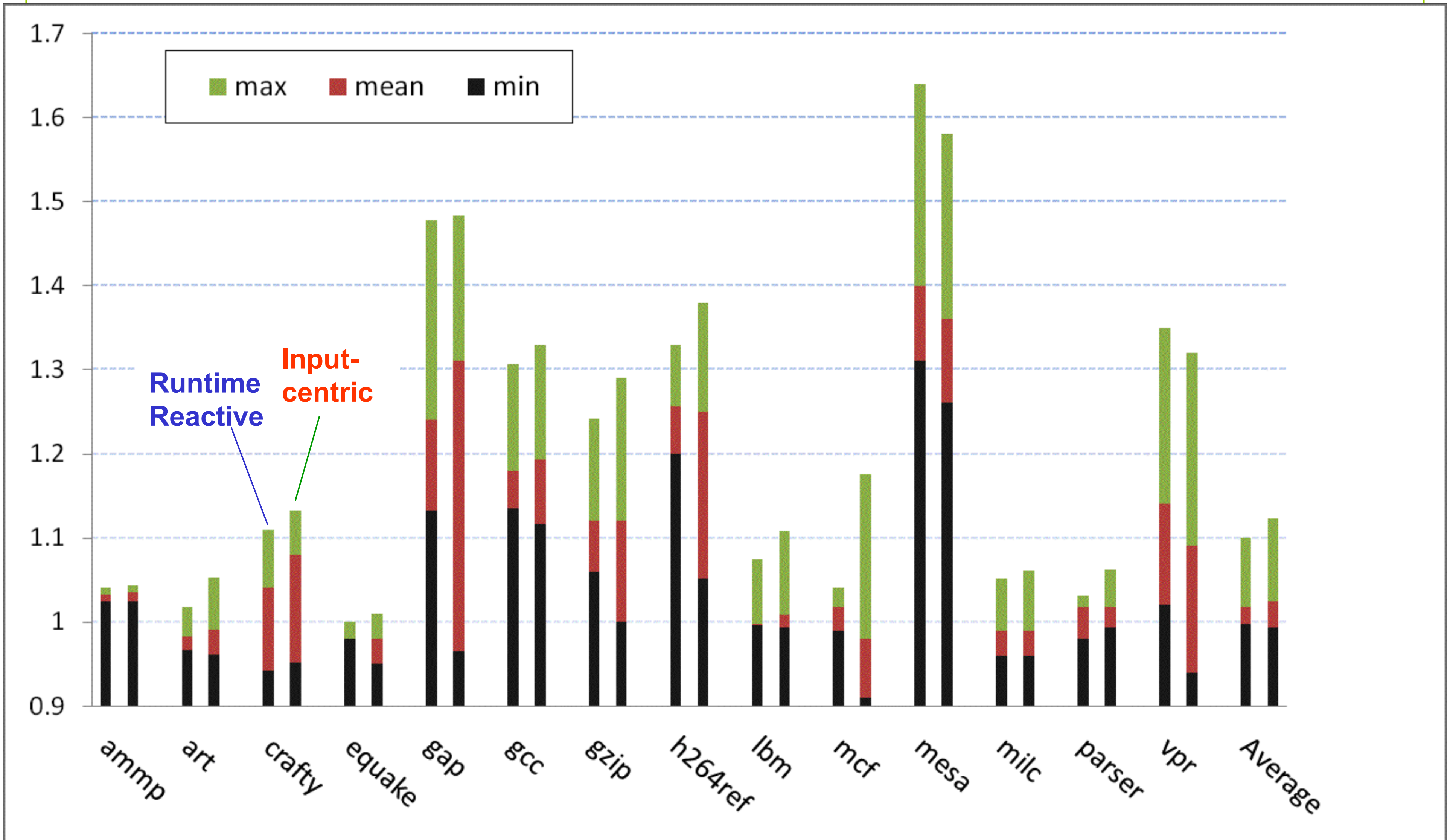
- Input-centric adaptation
  - Models from inputs to suitable versions
  - Predict the best version to run in a new execution
- Reactive approach [Chuang+:07]
  - Timing each version and use the best for the remaining execution

Experiment Setting

- Versions creation
  - IBM XL C compiler
  - 5 code versions from feedback-driven opt
- Machines
  - IBM Power5, AIX 5.3
- Benchmarks
  - 14 C programs from SPEC2000 & SPEC2006
- Inputs
  - 10--120
  - Some from University of Alberta (J. N. Amaral's group)
  - Others collected or created by us

Speedup by Version Selection

Baseline: static compilation at highest opt level



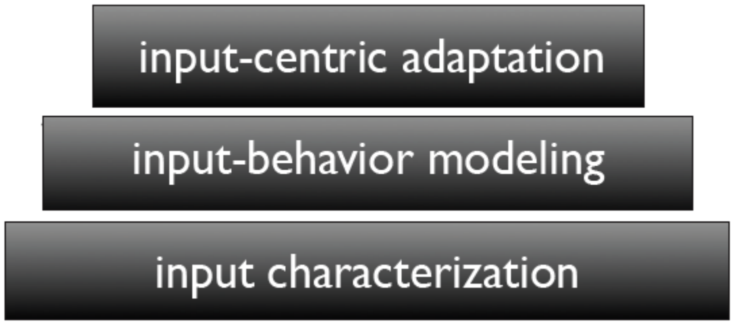
Discussions

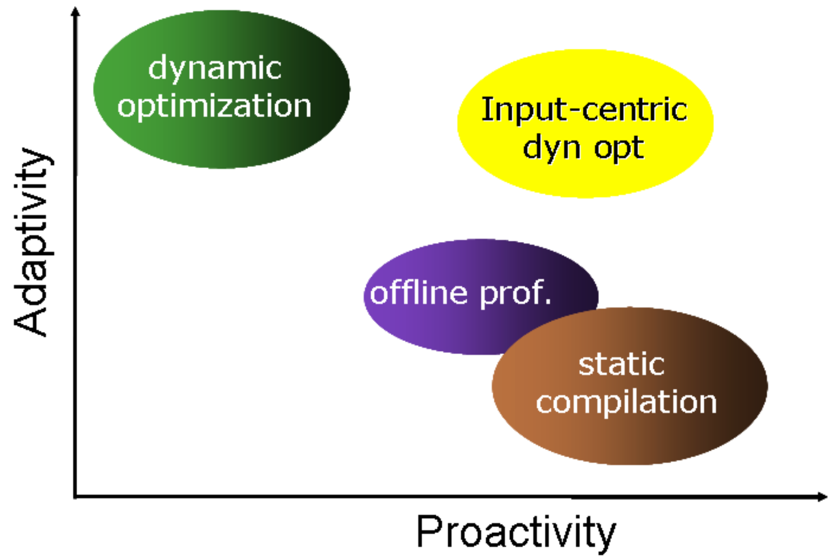
- Three steps for input-centric optimizations
  - Profile collection (offline)
  - Seminal beh recog. & input-beh model construction (offline)
  - Proactive behavior prediction & optimizations (online)
- Input-centric paradigm is fundamental
  - May benefit many other optimizations
    - Anywhere runtime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

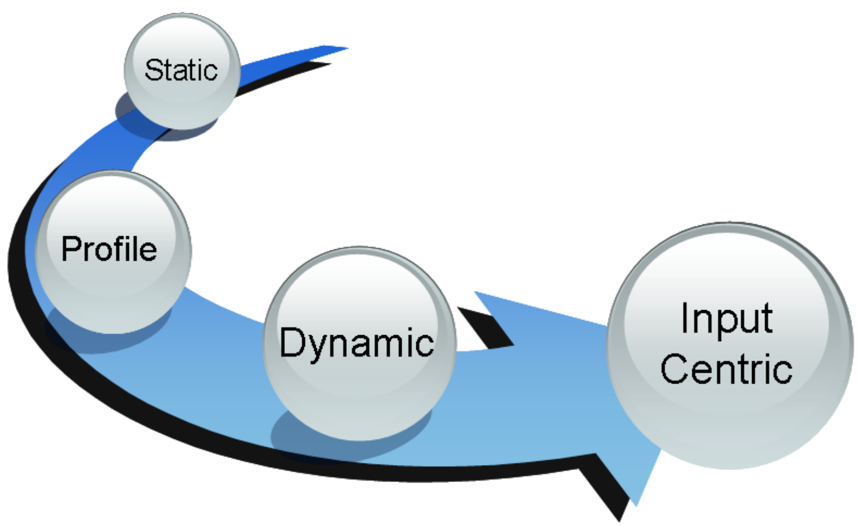
Related Work

- Phase-based adaptive recompilation
  - [Gu & Verbrugge: CGO'08]
- Benchmark design
  - [Berube & Amaral: SPEC'07]
- 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]

Conclusions



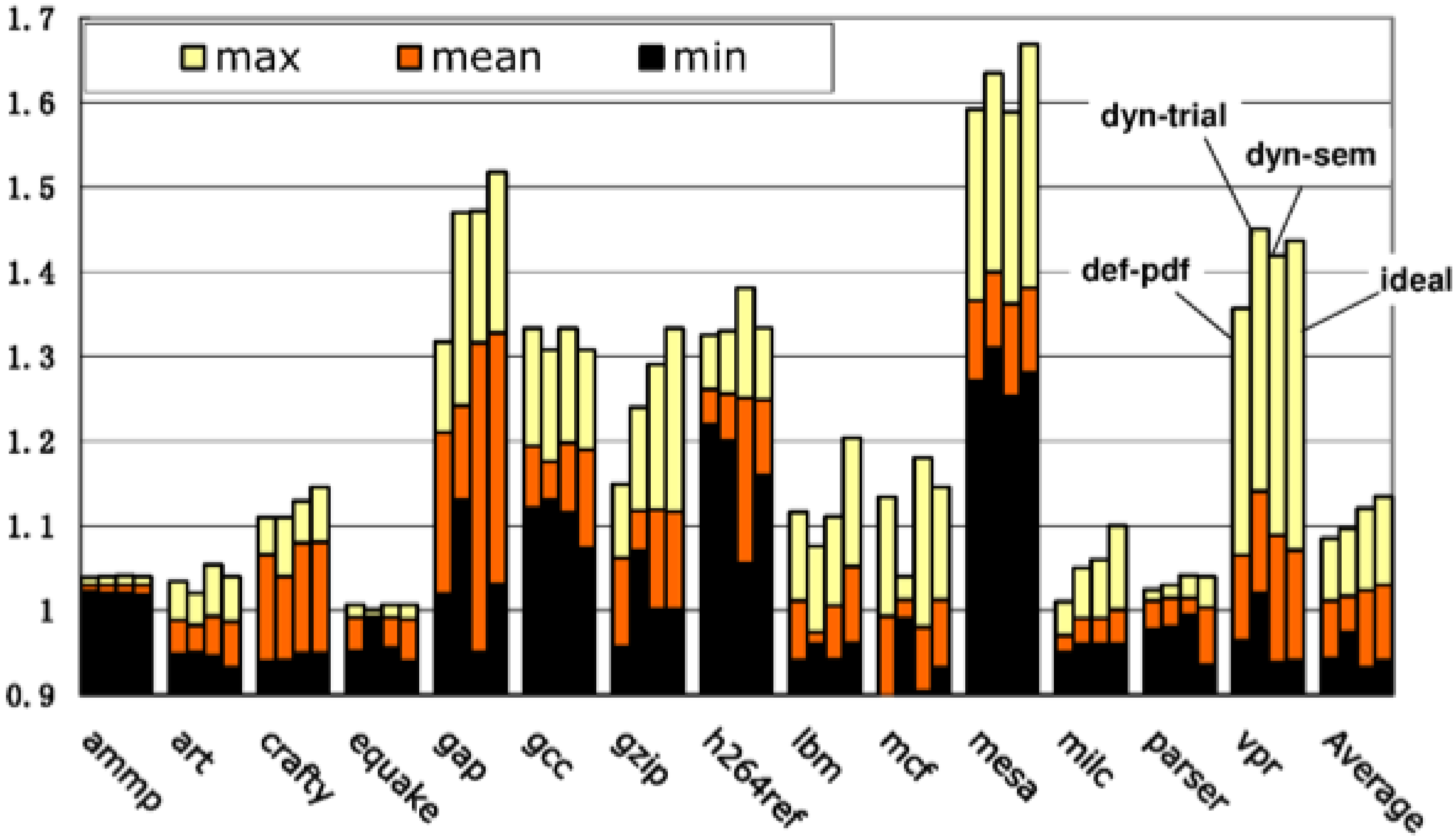




Thanks!

Potential Speedup in Version Selection

Baseline: static compilation at highest opt level



Name	# of lines of code	# of inputs	factors of changes caused by inputs	# of sem. beh.	accuracy edge	accuracy node	accuracy data
ammp	13263	20	9.9×10^1	1	100	91.1	99.7
art	1270	108	4.0×10^4	4	100	80.0	96.1
crafty	19478	14	4.6×10^8	2	90.8	44.5	79.3
equake	1513	100	1.0×10^2	1	100	96.3	99.3
gap	59482	12	1.1×10^8	7	56.3	69.7	88.5
gcc	484930	72	1.1×10^6	54	93.6	95.4	95.6
gzip	7760	100	4.3×10^7	6	83.5	69.0	94.5
h264ref	46152	20	2.1×10^9	4	97.0	97.8	99.7
lbm	875	120	6.0×10^6	3	100	100	100
mcf	1909	64	1.4×10^5	10	100	89.5	97.5
mesa	50230	20	2.0×10^1	1	99.5	12.2	100
milc	12837	10	2.1×10^9	18	100	52.0	99.7
parser	10924	20	2.1×10^6	2	79.2	78.0	90.8
vpr	16976	20	3.9×10^6	9	64.0	82.2	95.8