FlexJava: Language Support for Safe and Modular Approximate Programming

Jongse Park, Hadi Esmaeilzadeh, Xin Zhang, Mayur Naik, and William Harris

Approximate Programming

Programmer's **manual/explicit** specification
[EnerJ PLDI'11, Rely OOPSLA'13]

AUTOMATE Approximate computing

FlexJava
- Small set of language extensions
- Reducing annotation efforts
- Language-compiler co-design

FlexJava supports
- Safety
- Modularity
- Generality
- Scalability

Language Design

Safe Programming in FlexJava

Selectively approximating data and operations

```
int computeLuminance (float r, float g, float b) {
  float luminance = r * 0.3f + g * 0.6f + b * 0.1f;
  loosen(luminance);
  return luminance; }
```

Control flow safety

```
int fibonachi(int n) { int r;
  if (n < 1) r = n;
  else r = fibonachi(n - 1) + fibonachi(n – 2);
  loosen(r);
  return r; }
```

Predication to improve approximability

```
double sobel(double[][] p) {
  double x, y, g, r;
  x = p[0][0] + …;
  y = p[0][2] + …;
  g = sqrt(x * x + y * y);
  if(g > 0.7) r = 0.7;
  else r = g;
  loosen(r);
  return r;
}
```

```
double sobel(double[][] p) {
  double x, y, g, r;
  x = p[0][0] + …;
  y = p[0][2] + …;
  g = sqrt(x * x + y * y);
  r = (g > 0.7) ? 0.7 : g;
  loosen(r);
  return r;
}
```

Modular Approximate Programming

Scoped Approximation

```
int p = 1;
for (int i = 0; i < a.length; i++)
  p *= a[i];
for (int I = 0; I < a.length; i++) {
  p = b[i];
  loosen(p); }
```

Scope of approximation

Reuse and library support in FlexJava

```
static int square (int a) {
  int s = a * a;
  loosen(s); // enabled
  return s; }
public static void main
            (String[] args) {
  int x = 2 + square(3);
  loosen(x);
  System.out.println(x); }
```

```
static int square (int a) {
  int s = a * a;
  loosen(s); // disabled
  return s; }
public static void main
            (String[] args) {
  int x = 2 + square(3);
  System.out.println(x); }
```

Inter-procedural approximation: ***loosen_invasive***

```
static int square (int a) {
  int s = a * a;
  return s; }
public static void main
            (String[] args) {
  int x = 2 + square(3);
  loosen(x);
  System.out.println(x); }
```

```
static int square (int a) {
  int s = a * a;
  return s; }
public static void main
            (String[] args) {
  int x = 2 + square(3);
  loosen_invasive(x);
  System.out.println(x); }
```

Generality in FlexJava: Support for Coarse Grained Approximation

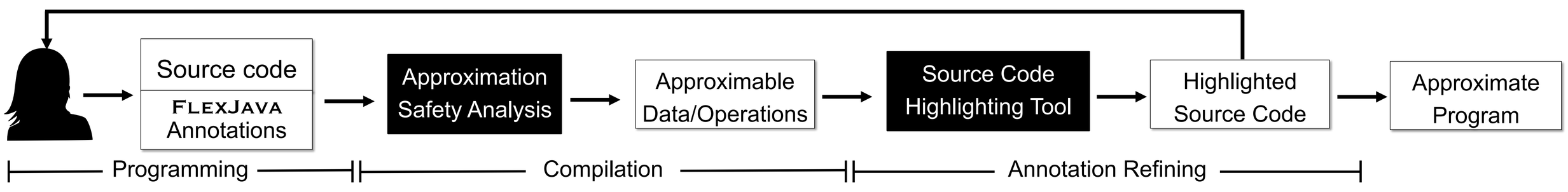
Loop perforation

```
begin_loose ("PERFORATION", 0.10);
  for (int I = 0; I < n; i++) { ... }
end_loose ();
```

Neural acceleration

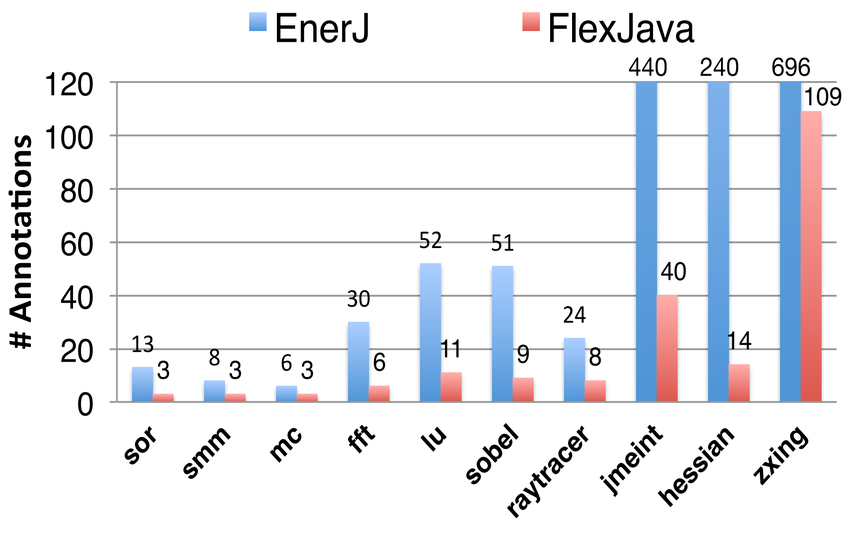
```
Double foo (Double x, Double y) {
  begin_loose ("NPU", x, y);
    p = Math.sin (x) + Math.cos (y);
    q = 2 * Math.sin (x + y);
  end_loose (p, q);
  return p + q;
```

Programming Workflow

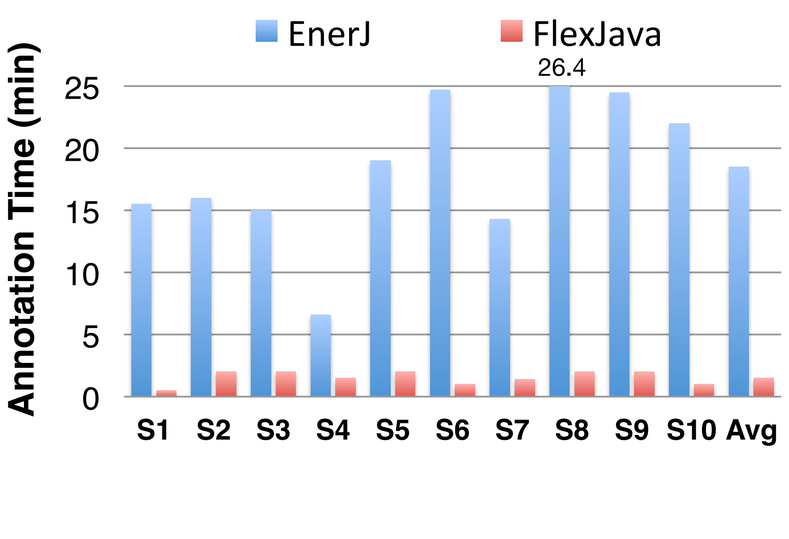


Evaluation

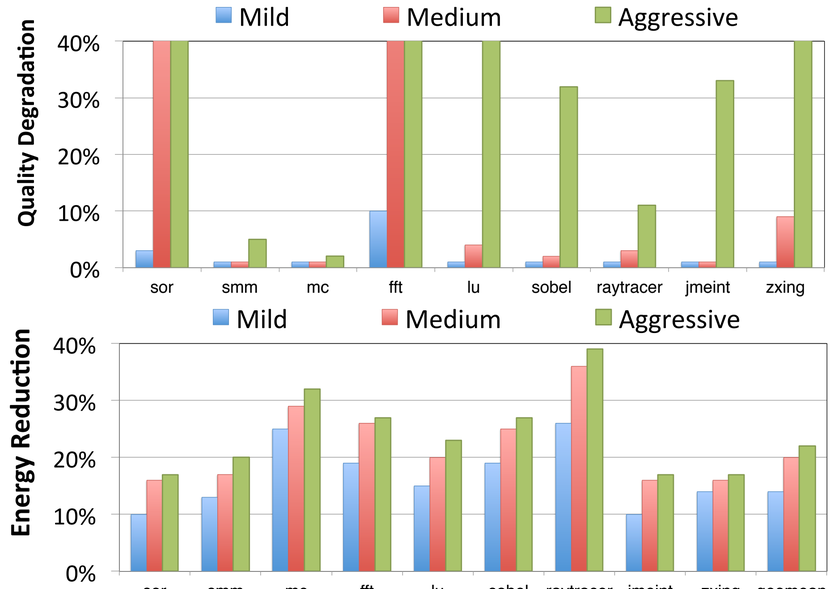
[1] Number of annotations: Comparison with a leading approximate programming language – EnerJ



[2] Programmer effort (Annotation time): Comparison with a leading approximate programming language – EnerJ



[3] Energy Reduction and Speedup: Fine-grained approximation (granularity of data and operations)



Alternative Computing Technologies (**ACT**) Lab – Georgia Institute of Technology