Combining Execution Modes of LLVM Bitcode on GraalVM

Christoph Pichler
Johannes Kepler University
christoph.pichler@jku.at

Paley Li
Oracle Labs
paley.li@oracle.com

Roland Schatz
Oracle Labs
roland.schatz@oracle.com

Hanspeter Mössenböck
Johannes Kepler University
hanspeter.moessenboeck@jku.at

Why Not Only Native Execution?

Python/JS... pseudo code
p = Point(x: 4, y: -3)
diff = calcDiff(p)

code
double calcDiff(struct Point *p) {
    return sqrt(p->x*p->x+p->y*p->y);
}

passing managed (e.g. JS) object to C code

Why Not Only Managed Execution?

● Slow warm-up for managed execution in GraalVM (due to dynamic compilation)
● Thus: Less managed/more native code improves (warm-up) performance

Our Approach: Hybrid Execution

Managed Execution
● High peak performance
● Polyglot applications

Native Execution
● High warm-up performance

Current State

Benchmark: LXML parser in Python, which uses libxml in C
● Interquartile range: 0.02x

Different tasks show different results
→ performance highly depends on decision which function to execute natively or via GraalVM
● Restrictions in current state
  ○ Manual decision how a callee is run (GraalVM/native)

References