Analysis and Optimization of Complex Software Systems

Costin Iancu, Wim Lavrijsen
Lawrence Berkeley National Laboratory

Koushik Sen
University of California at Berkeley
As every program becomes parallel, conventional compilers address only parts of software development concerns. We need novel approaches able to handle complex modular parallelism.

Our conjecture: complex runtime parallel behavior is best handled by dynamic program analysis
Programming Complex Systems

- **High Level Languages (HLL), application frameworks are pervasive**
  - Domain Specific Language: e.g. Tensors (TCE) in HPC, Pig Latin in commercial
  - Frameworks. e.g. Trilinos or Hadoop
  - HLL proper: Scala, Python, Matlab, R

- **No good optimizing compiler/SDK exists for (parallel) HL concepts**
  - Significant engineering challenges
  - Static compilation is inherently limited when dealing with complex dynamic behavior
  - Semantic gap between HL and hardware/system
Dynamic Analysis in Practice

- Dynamic Analysis (DA): run -> learn -> transform -> check -> repeat

- Many success stories using Dynamic Analysis
  - Program verification, testing, bug detection
  - Speculative synchronization and parallelism transformations
  - Runtime code optimization (JIT) (e.g. javascript, PyPy)

- DA handles multi-language, multi-paradigm, multi-domain applications
  - Relatively short development cycle for powerful analyses
  - Easy to specialize for the problem domain/infrastructure

- DA adds runtime overhead, sometimes prohibitive
  - Need to monitor program execution and infer behavior
<table>
<thead>
<tr>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soundness</td>
<td>Soundy (?)</td>
</tr>
<tr>
<td>Precision</td>
<td>Conservative, many false positives</td>
</tr>
<tr>
<td>Coverage</td>
<td>Whole program</td>
</tr>
<tr>
<td>Usage</td>
<td>Automated</td>
</tr>
<tr>
<td>Language target</td>
<td>Static languages, fixed syntax</td>
</tr>
<tr>
<td>Analyses Type</td>
<td>Language specific, domain independent</td>
</tr>
<tr>
<td>Domains</td>
<td>Best for single core code generation</td>
</tr>
</tbody>
</table>
Hybrid Analyses

- Use compiler as instrumenter to lower runtime overhead
  - Static analysis to guide the dynamic analysis

- Provide composable domain/language independent building blocks
  - E.g. event hooks, annotation, and instrumentation

- Provide for domain/language specialization

- Provide “generic” canonical analyses that can be composed
  - E.g. data race detection, order analysis

- Provide presentation layers: translate machine/runtime info into English – needs focus