Fast, Effective Dynamic Compilation

Joel Auslander, Craig Chambers, Susan Eggers, Brian Grant,
Markus Mock, Matthai Philipose

April 19, 1996

Dynamic Compilation

What is it?

Defer some compilation to run time
Optimize code on the basis of **runtime constants**

Why do it?

Better performance
Many interesting applications have runtime constants

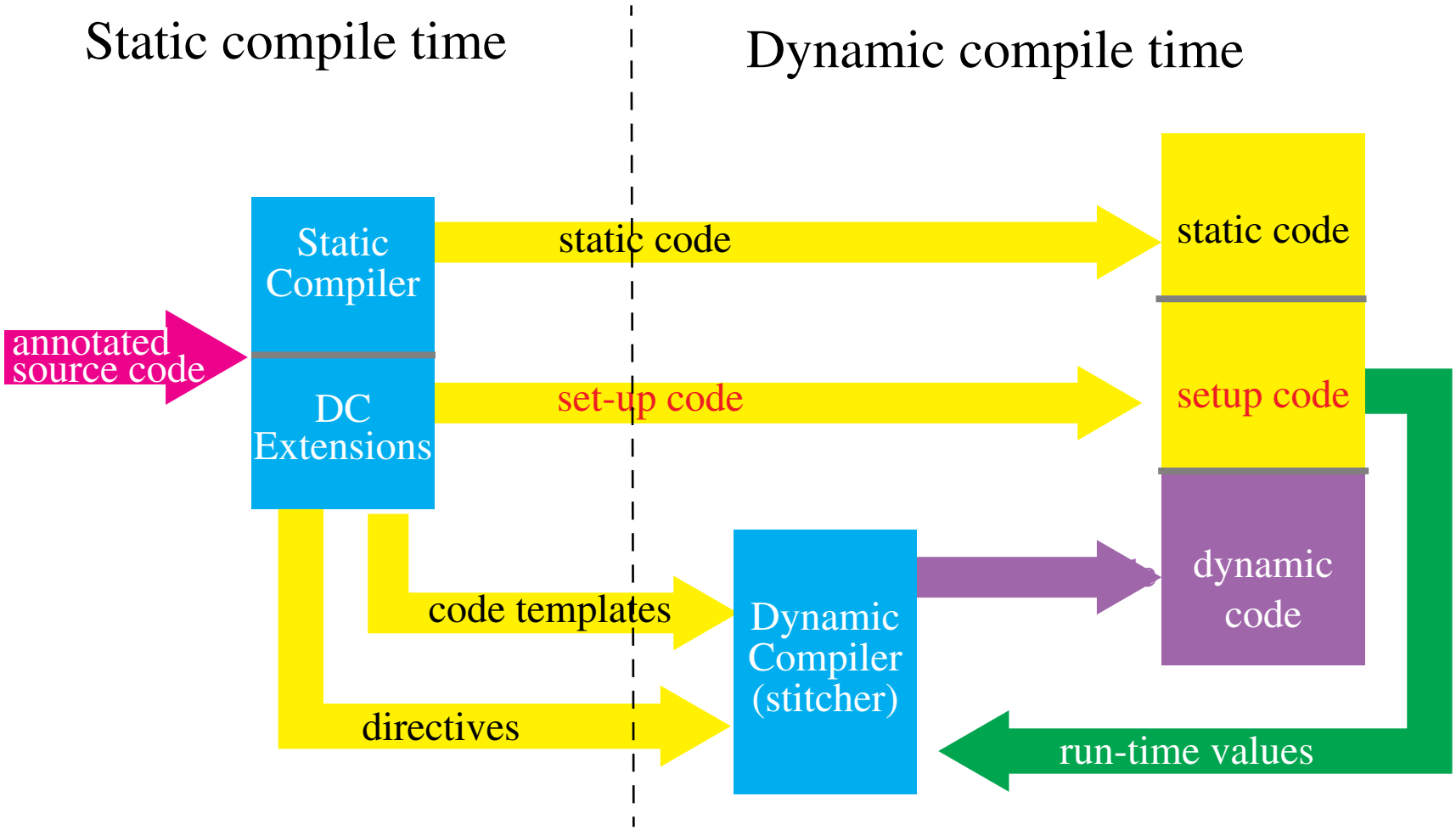
Why is it difficult?

Tradeoff: *fast dynamic compile time* vs. *higher code quality*

Our solution:

Do maximum work at static compile time

Dynamic Compiler Framework



Example: Cache Lookup

```
cacheResult cacheLookup(void *addr, Cache *cache) {

  unsigned blockSize = cache->blockSize;
  unsigned numLines = cache->numLines;
  unsigned tag = (unsigned) addr / (blockSize * numLines);
  unsigned line = ((unsigned) addr / blockSize) % numLines;
  setStructure *setArray = cache->lines[line];
  unsigned associativity = cache->associativity;
  int set;

            for (set = 0; set < associativity; set++) {
   if (setArray[set].tag == tag) return CacheHit;
  }
  return CacheMiss;

}
```

Cache Lookup Example: Template Code

```
cacheResult cacheLookup(void *addr, Cache *cache) {

 dynamicRegion {

  unsigned tag = (unsigned) addr /    ;
  unsigned line = ((unsigned) addr /    ) %    ;
  setStructure *setArray =    [line];



  unrolled for (set) {
    if (setArray[    ].tag == tag) return CacheHit;

  }
  return CacheMiss;

 }
```

Cache Lookup Example: End Result

4-way associative, 32 byte cache block

```
unsigned tag = (unsigned)addr >> 14;
unsigned index = ((unsigned)addr >> 5) & 511;
setStructure *setArray = cacheLines[index];
if (setArray[0].tag == tag) return CacheHit;
if (setArray[1].tag == tag) return CacheHit;
if (setArray[2].tag == tag) return CacheHit;
if (setArray[3].tag == tag) return CacheHit;
return CacheMiss;
```

Dynamic optimizations:

- constant propagation
- constant folding
- strength reduction
- complete loop unrolling

Preliminary Results

Benchmark	Run-time Constant Configuration	Asymptotic Speedup	Break-even Point
Reverse-polish stack-based desk calculator	polynomial expression in x & y	1.7	916 interpretations
Scalar-matrix multiply	the scaling factor	1.6	31K multiplies
Sparse matrix-vector multiply	200x200 matrix, 10 elements/row, 5% density	1.8	2650 matrix multiplies
	96x96 matrix, 5 elements/row, 5% density	1.5	1850 matrix multiplies
Event dispatcher in SPIN	7 predicate types, 10 event guards	1.4	722 dispatches
Quicksort record sorter	4 keys	1.2	3050 records

Future Work

Faster dynamic compile time:

- Merge set-up code, stitcher directives and stitcher
- Lazy vs. eager stitching

Expand repertoire of optimizations:

- Register allocation at runtime
- Scheduling between unrolled loop iterations

Handling more applications:

- Runtime constants that can change
- Inlining
- Multi-way loop unrolling

The Holy Grail:

- Automatic identification of dynamic regions & runtime constants

```
 dynamicRegion(cache) {

  unrolled

  }

 }
}
```

Example: Cache Lookup

```
cacheResult cacheLookup(void *addr, Cache *cache) {

  unsigned blockSize = cache->blockSize;
  unsigned numLines = cache->numLines;
  unsigned tag = (unsigned) addr / (blockSize * numLines);
  unsigned line = ((unsigned) addr / blockSize) % numLines;
  setStructure *setArray = cache->lines[line];
  unsigned associativity = cache->associativity;
  int set;

 for (set = 0; set < associativity; set++) {
   if (setArray[set].tag == tag) return CacheHit;
  }
  return CacheMiss;

}
```

Cache Lookup Example with Annotations

```
cacheResult cacheLookup(void *addr, Cache *cache) {

dynamicRegion(cache) {
  unsigned blockSize = cache->blockSize;
  unsigned numLines = cache->numLines;
  unsigned tag = (unsigned) addr / (blockSize * numLines);
  unsigned line = ((unsigned) addr / blockSize) % numLines;
  setStructure *setArray = cache->lines[line];
  unsigned associativity = cache->associativity;
  int set;

 unrolled for (set = 0; set < associativity; set++) {
   if (setArray[set].tag == tag) return CacheHit;
  }
  return CacheMiss;
 }
}
```