# SPUR

A TRACE-BASED JIT COMPILER FOR .NET (MAKES JAVASCRIPT SUPER FAST)

# Nikolai Tillmann Microsoft Research

http://research.microsoft.com/Spur/

#### Introduction

- Spur: One runtime + JIT for all languages on .NET!
- Tracing JIT, advanced optimizations
- Platform for research experiments

# **Background: JavaScript**

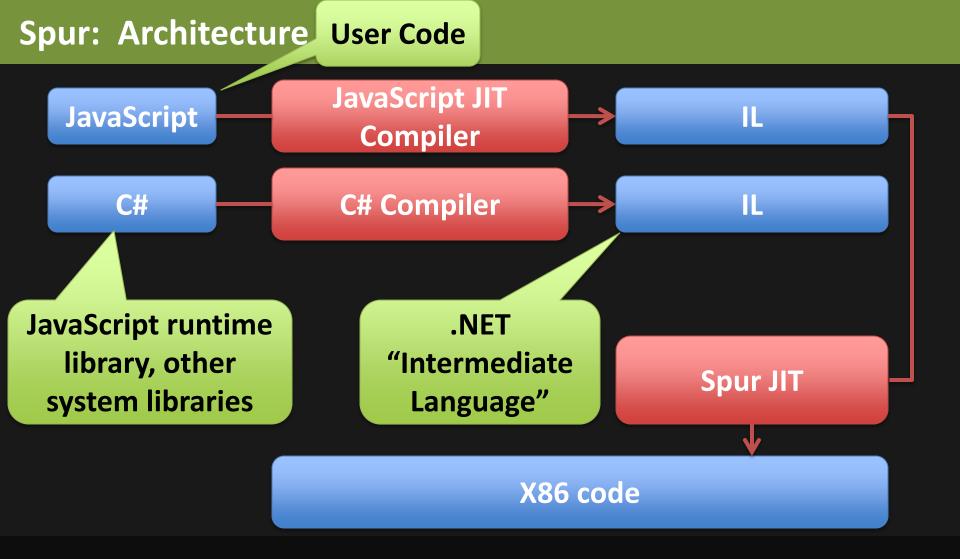
```
var sum = 0
for (var i = 0; i < 1000; i++) {
    if (i == 990) {
        sum += " Hello World "
    }
    sum += 1
}
print(sum)
result: "990 Hello World 111111111"</pre>
```

- Dynamically typed, type inference is difficult in the general case
- The semantics of "add" depend on its operands, this results in lots of runtime checks

# **Basic idea: Trace Compilation**

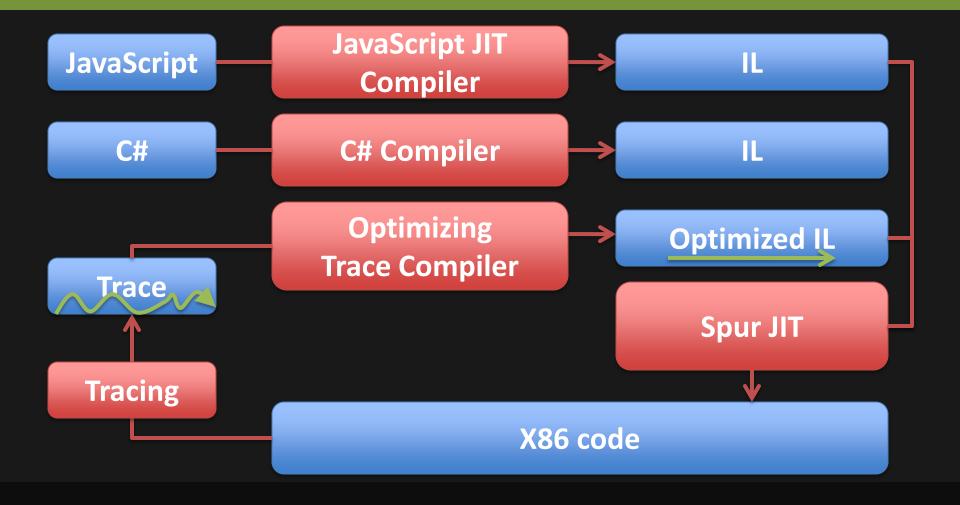
```
var sum = 0
for (var i = 0; i < 1000; i++) {
    if (i == 990) {
        sum += " Hello World "
    }
    sum += 1
}
print(sum)
result: "990 Hello World 111111111"</pre>
```

- Trace compilation of loop for case where "sum" is an integer
- After "sum" becomes a string, execution is resumed in un-optimized code



- Java Script source compiled to IL, references JavaScript Runtime written in C#
- Compiled Script and JavaScript Runtime are JITted to x86 code

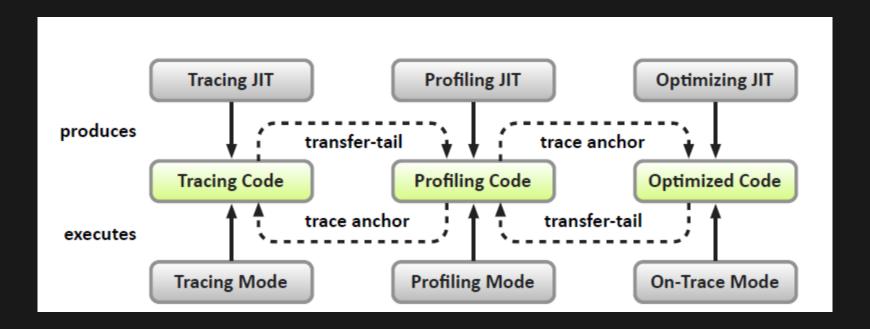
# **Spur: Architecture**



Spur traces and optimizes the combination of

- the JavaScript IL Code and the
- JavaScript Runtime IL Code

#### **Runtime: Code Version**



#### **Spur initially generates Tracing Code:**

simple JIT with loop counters, only block-level register allocation

If loop is hot, it generates Profiling Code:

same as tracing code, but with call backs to monitor trace

If execution stays on trace, it generates Optimized Code

implements all standard (and some speculative) optimizations

# Which means, that we go from this ...

```
for (var n = 0; n < 1000; n++) {
    for (var n2 = 0; n2 < 1000; n2++) {
        for (var i = 0; i < a.length - 1; i++) {
            var tmp = a[i];
            a[i] = a[i + 1];
            a[i + 1] = tmp;
        }
    }
}</pre>
```

**⊙** 35 method calls, 129 guards, 224 total instructions

#### to this ...

```
tic void SetObjectAndGetValue(Reference r, ref ValueWrapper obj, ref ValueWrapper result, FunctionInstance currentFunction)
                                                                                                                                                                                          ublic override void GetValueInto(ObjectInstance instance, ref ValueWra public sealed override ve if (this.obj == null)
                                                                                                                                                                                                                                                                                                                                          void SetValueInto(ref ValueWrapper wrapper) {
   if (obj.typeCode == TypeCode.Object) {
                                                                                                                                                                                          ArrayInstance arrayInstance = (ArrayInstance)instance;
                                                                                                                                                                                                                                                                                                                                            eption(JSError.ObjectExpected, this.objectContext);
          typedObject = obj.wrappedObjectInstance;
                                                                                      public sealed override void petValueInto(ref ValueMrapper wrapper) {
if (this.obj == null)
                                                                                                                                                                                          result.wrappedObject = null;
      if (typedObject != r.obj)
                                                                                                                                                                                                                                                                                                                                              property = this.GetProperty(ref ob);
                                                                                                                                                                                                                                                                                                                    'ypedObjectProperty/*?*/
Lf (property != null) {
       r.obj = typedObject;
                                                                                                                                                                                          result.wrappedDouble = arrayInstance.len:
                                                                                                               cception(JSError.ObjectExpected, this.objectContext);
                                                                                                                                                                                                                                                                                                                     property.GetValueInto(ob,
                                                                                                                                                                                          result.typeCode = TypeCode.Double;
                                                                                        ObjectInstance ob = this.obj
    r.obj = Convert.ToObject(ref obj, currentFunction.Globals)
                                                                                                                 / property = this.GetProperty(ref ob);
                                                                                         /pedObjectProperty/*?*/ property = this.
f (property != null) {
  property.GetValueInto(ob, ref wrapper);
  r.GetValueInto(ref result);
    lic sealed override void SetValueInto(ref ValueNrapper wrapper) (
(this.obj == null)
                                                                                                                                                                               olic override void GetValueInto(ObjectInstance instance, ref ValueWrapper result) ( s | r mil) (
                                                                                                                                                                              ArrayInstance arrayInstance = (ArrayInstance)instance;
                                                                                                    ructor reCons = this.obj as RegExpConstructor;
                                                                                                                                                                             result.wrappedObject = null;
                           eption(JSError.ObjectExpected. this.objectContext):
                                                                                         f (recons != null)
wrapper.SetValue(recons.GetPropertyValue(this.name));
                                                                                                                                                                                                                                                                                                                   oblic sealed override void petValueInto(ref ValueKrapper wrapper) (
if (this.obj == null)
                                                                                                                                                                             result.wrappedDouble = arrayInstance.len;
  TypedObjectProperty/*?*/ property = this.GetProperty(ref ob);
if (property != null) {
                                                                                                                                                                             result.typeCode = TypeCode.Double;
                                                                                         wrapper.Clear();
    property.GetValueInto(ob, ref wrapper);
                                                                                                                                                                                                                                                                                                                   throw new JScriptException(J
ObjectInstance ob = this.obj;
                                                                                                                                                                                                                                                                                                                                           ception(JSError.ObjectExpected, this.objectContext)
                                                                                             / property = this.GetProperty(ref ob);
  RegExpConstructor reCons = this.obj as RegExpConstructor; if (reCons != null)
                                                                                          /^ requires this.obj != null;
                                                                                                                                                                                       internal virtual void petValueAtIndexInto(ref ValueWrapper result, int index) {
  var i = (uint)index;
                                                                                                                                                                                                                                                                                                                     f (property != null) {
 property.GetValueInto(ob, ref wrapper);
    wrapper.SetValue(reCons.GetPropertyValue(this.name));
                                                                                         TypedObjectProperty/*?*/ property = this.GetLocalTypedProperty();
                                                                                                                                                                                                 t sparseValue = null;
                                                                                          (property != null) return property;
serenceToProperty/*?*/ prototypeBinding = this.GetPrototypeBinding();
                                                                                                                                                                                           if (i < int.MaxValue && this.sparseArray != null && this.sparseArray.TryGetValue
    wrapper.Clear();
                                                                                                                                                                                                                                                                                                                     egExpConstructor reCons = this.obj as RegExpConstructor;
f (reCons != null)
                                                                                                                                                                                             result.SetValue(sparseValue);
                                                                                             le (prototypeBinding != null) {
                                                                                                                                                                                                                                                                                                                      wrapper.SetValue(reCons.GetPropertyValue(this.name));
    vate TypedObjectProperty/*?*/ petProperty(ref ObjectInstance or
'/^ requires this.obj != null;
                                                                                                                                                                                          } else if (index >= 0) {
                                                                                          ob = ob.Prototype;
                                                                                                                                                                                             ObjectInstance/*?*/ prototype = this.Prototype as ObjectInstance;
if (prototype != null)
                                                                                                                                                                                                                                                                                                                     wrapper.Clear();
                                                                                          property = prototypeBinding.GetLocalTypedProperty();
                                                                                          if (property != null) return property;
prototypeBinding = prototypeBinding.GetPrototypeBinding();
   TypedObjectProperty/*?*/ property = this.GetLocalTypedProperty(
if (property != null) return property;
ReferenceToProperty/*?*/ prototypeBinding = this.GetPrototypeBi
                                                                                                                                                                                               prototype.GetValueAtIndexInto(ref result, index);
                                                                                                                                                                                                                                                                                                                                          void betValueInto(ref ValueWrapper wrapper) {
                                                                                                                                                                                                                                                                                                                   if (this.obj == null)
                                                                                                                                                                                                                                                                                                                   throw new JScriptException(J
ObjectInstance ob = this.obj;
                                                                                                                                                                                                                                                                                                                                            eption(JSError.ObjectExpected, this.objectContext)
                                                                                                                                                                                                result.Clear();
     ile (prototypeBinding != null) {
                                                                                                                                                                                                                                                                                                                    TypedObjectProperty/*?*/ property = this.@
if (property != null) {
   property.GetValueInto(ob, ref wrapper);
                                                                                                                                                                                                                                                                                                                                            / property = this.GetProperty(ref ob);
     ob = ob.Prototype;
     property = prototypeBinding.GetLocalTypedProperty();
                                                                                                                                                                                             this.GetPropertyValueInto(this.Globals.nameTable.GetNameFor(Convert.ToString(i
                                                                                      internal object/*?*/ GetPropertyValue(IName nam
    if (property != null) return property;
prototypeBinding = prototypeBinding.GetPrototypeBinding();
                                                                                          ValueWrapper wrapper = new ValueWrapper();
                                                                                                                                                                                                                                                                                                                              structor reCons = this.obj as RegE
                                                                                          this.GetPropertyValueInto(name, ref wrapper);
                                                                                                                                                                                            id SetValueInto(ref ValueWrapper wrapper) {
                                                                                          return wrapper.BoxedValue();
                                                                                                                                                                                                                                                                                                                    ception(JSError.ObjectExpected, this.objectContext);
                                                                                                                                                                                                                                                                                             throw new JScriptException(
ObjectInstance ob = this.obj;
    TypeObject obType = this.obj.Type:
                                                                                                                                                                                                                                                                                                                       / property = this.GetProperty(ref ob);
                                                                                                                                                                                                                                                                                              f (property |= null) {
                                                                                                                                                                                                                                                                                                                 lic sealed override void SetValueInto(ref ValueWrapper wrapper) {
f (this.obj == null)
                                                                                                                                                                                             (obType == this.last_type) {
                                                                                        TypeObject obType = this.obj.Type;
                                                                                                                                                                                            prop = this.property;
    ypedObjectProperty prop;
if (obType == this.last_type) {
                                                                                                                                                                                               (prop != null) retu
                                                                                                                                                                                                                                                                                                                throw new JScriptException(JSError.ObjectExpected, this.objectContext);
ObjectInstance ob = this.obj;
                                                                                                                                                                                             if (obType == WithScope.WithType)
                                                                                        TypeObject obType = this.obj.Type;
    prop = this.property;
if (prop != null) return prop;
if (obType == WithScope.WithType)
                                                                                                                                                                                                                                      pe)this.obi).GetLocalProperty(this.name):
                                                                                                                                                                                              this.property = prop = ((Wi
                                                                                                                                                                                                                                                                                             if (reCons != n
                                                                                                                                                                                                                                                                                                                                         / property = this.GetProperty(ref ob):
                                                                                          /pedObjectProperty prop;
{ (obType == this.last_type) {
                                                                                                                                                                                                    f (obType.IsMutable)
                                                                                                                                                                                                                                                                                                                  property.GetValueInto(ob, ref wrapper);
                                                                                                                                                                                              this.property = prop = obType.GetLocalProperty(this.name);
                                                                                         prop = this.property;
                                                                                                                                                                                                                                                                                              wrapper.Clear
       this.property = prop = ((Wi
                                              ope)this.obj).GetLocalProperty(t
                                                                                                                                                                                                   n prop:
                                                                                           f (prop != null) return prop;
f (obType == WithScope.WithType)
            if (obType.IsMutable)
                                                                                                                                                                                                                                                                                                                 RegExpConstructor reCons = this.obj as RegExpConstructor;
if (reCons l= null)
      this.property = prop = obType.GetLocalProperty(this.name):
                                                                                                                                                                                          this.last_type = obType;
                                                                                            this.property = prop = ((WithScope)this.obj).GetLocalProperty(this.name);
         ırn prop;
                                                                                                                                                                                            f (obType == WithScope.WithType)
                                                                                               if (obType.IsMutable)
                                                                                                                                                                                                                                                                                                                 wrapper.SetValue(reCons.GetPropertyValue(this.name));
                                                                                                                                                                                             this.property = prop = ((WithScope)this.obj).GetLocalProperty(this.name);
                                                                                           this.property = prop = obType.GetLocalProperty(this.name);
  this.last_type = obType;
                                                                                                                                                                                                                                                                                                                 wrapper.Clear():
                                                                                                                                                                                            this.property = prop = obType.GetLocalProperty(this.name);
                                                                                                                                                                                                                                                                                              f (property != }
    this.property = prop = ((With
                                         cope)this.obj).GetLocalProperty(thi
                                                                                        this.last_type = obType;
                                                                                                                                                                                                                                                                                              property.GetValueInto(ob, ref wrapper);
                                                                                         f (obType == WithScope.WithType)
    this.property = prop = obType.GetLocalProperty(this.name);
                                                                                         this.property = prop = ((WithScope)this.obj).GetLocalProperty(this.name);
                                                                                                                                                                                                                   oid GetValueAtIndexInto(ref ValueWrapper result,
                                                                                                                                                                                                                                                                                             RegExpConstructor reCons = this.obj as RegExpConstructor;
if (reCons l= null)
                                                                                                                                                                                           var i = (uint)index;
                                                                                         this.property = prop = obType.GetLocalProperty(this.name);
                                                                                                                                                                                                 t sparseValue = null;
                                                                                                                                                                                                                                                                                              wrapper.SetValue(reCons.GetPropertyValue(this.name));
internal object/*?*/ GetPropertyValue(INa,
                                                                                                                                                                                           if (i < int.MaxValue && this.sparseArray != null && this.sparseAr
                                                                                                                                                                                                                                                                                              wrapper.Clear();
    ValueWrapper wrapper = new ValueWrapper();
                                                                                                                                                                                             result.SetValue(sparseValue);
                                                                                                                            void SetValueInto(ref ValueWrapper wrapper) {
                                                                                                                                                                                                                                                                                                               ublic sealed override void betValueInto(ref ValueMrapper wrapper) (
if (this.obj == null)
                                                                                                                                                                                          } else if (index >= 0) {
    this.GetPropertyValueInto(name, ref wrapper);
                                                                                                     throw new JScriptException(JSError.ObjectExpected, this.objectContext);
ObjectInstance ob = this.obj;
                                                                                                                                                                                             ObjectInstance/*?*/ prototype = this.Prototype as ObjectInstance;
                                                                                                                                                                                                                                                                                                                throw new JScriptException(JSError.ObjectExpected, this.objectContext);
ObjectInstance ob = this.obj;
    return wrapper.BoxedValue();
                                                                                                                                                                                              if (prototype != null)
                                                                                                      ypedObjectProperty/*?*/ property = this.d
if (property != null) {
  property.GetValueInto(ob, ref wrapper);
                                                                                                                               / property = this.GetProperty(ref ob);
                                                                                                                                                                                                prototype.GetValueAtIndexInto(ref result, index);
                                                                                                                                                                                                                                                                                                                                         / property = this.GetProperty(ref ob);
                                                                                                                                                                                                                                                                                                                  property.GetValueInto(ob, ref wrapper);
                                                                                                                                                                                                result.Clear();
internal virtual void petValueAtIndexInto(ref ValueWrapper result,
  var i = (uint)index;
                                                                                                                                                                                                                                                                                                                  ngExpConstructor reCons = this.obj as RegExpConstructor;
(reCons != null)
                                                                                                                                                                                             this.GetPropertyValueInto(this.Globals.nameTable.GetNameFor(Convert.ToString
         ct sparseValue = null;
                                                                                                      wrapper.SetValue(reCons.GetPropertyValue(this.name));
  if (i < int.MaxValue && this.sparseArray != null && this.sparseAr
                                                                                                                                                                                                                                                                                                                  wrapper.SetValue(reCons.GetPropertyValue(this.name));
      result.SetValue(sparseValue);
                                                                                                                                                                                                                                                                                                                 wrapper.Clear():
  } else if (index >= 0) {
                                                                                                                                                                                                                        public sealed override void petValueInto(ref ValueWrapper wrapper) {
   if (this.obj == null)
     ObjectInstance/*?*/ prototype = this.Prototype as ObjectInstance;
                                                                                                            internal object/*?*/ GetPropertyValue(IName name) {
                                                                                                                                                                                                                                                                                                        vate TypedObjectProperty/*?*/ GetLocalTypedProperty()
/^ requires this.obj != null;
      if (prototype != null)
                                                                                                                ValueWrapper wrapper = new ValueWrapper();
        prototype.GetValueAtIndexInto(ref result, index);
                                                                                                                                                                                                                                                    property = this.GetProperty(ref ob):
                                                                                                                 this.GetPropertyValueInto(name, ref wrapper);
                                                                                                                                                                                                                                                                                                       TypeObject obType = this.obj.Type;
                                                                                                                                                                                                                            property.GetValueInto(ob, ref wrapper);
                                                                                                                 return wrapper.BoxedValue();
                                                                                                                                                                                                                                                                                                       if (obType == this.last_type) {
                                                                                                                                                                                                                                                                                                           c) = this.property;
[prop != null) return prop;
[obType == WithScope.WithType]
      this.GetPropertyValueInto(this.Globals.nameTable.GetNameFor(Convert.Tc)
                                                                                                                                                                                                                 ate TypedObjectProperty/*?*/ SetProperty(ref ObjectInstance ob)
                                                                                                                                                                                                              //^ requires this.obj |= null;
                                                                                                                     internal object/*?*/ GetPropertyValue(INam {
                                                                                                                                                                                                                                                                                                            iis.property = prop = ((With
if (obType.IsMutable)
                                                                                                                                                                                                                                                                                                                                                cope)this.obj).GetLocalProperty(this.name);
                                                                                                                                                                                                                 ublic override void GetValueInto(ObjectInstance instance, ref Value
                                                                                                                        ValueWrapper wrapper = new ValueWrapper(
                                                                                                                                                                                                                                                                                                            iis.property = prop = obType.GetLocalProperty(this.name);
   ArrayInstance arrayInstance = (ArrayInstance)instance;
                                                                                                                        this.GetPropertyValueInto(name, ref wrap
                                                                                                                                                                                                                   le (prototypeBinding != null) {
   result.wrappedObject = null;
                                                                                                                         return wrapper.BoxedValue();
                                                                                                                                                                                                                 ob = ob.Prototype;
                                                                                                                                                                                                                                                                                                             last_type = obType;
   result.wrappedDouble = arrayInstance.len;
                                                                                                                                                                                                                property = prototypeBinding.GetLocalTypedProperty();
                                                                                                                                                                                                                                                                                                            i.property = prop = ((WithScope)this.obj).GetLocalProperty(this.name);
                                                                                                                                                                                                                if (property != null) return property;
prototypeBinding = prototypeBinding.GetPrototypeBinding();
   result.typeCode = TypeCode.Double;
                                                                                                                                                                                                                                                                                                            i.property = prop = obType.GetLocalProperty(this.name);
```

# back to IL code equivalent to this C# code

```
double index = 0;
/* ... lots of hoisted code */
while (true) {
   if (index >= arrayIndex - 1) {
        // transfer
    if (index + 1 >= validArrayIndex) {
        // transfer
    /* bounds checks have been proven */
    var temp = array[(int) index];
    array[(int)index] = array[(int)index + 1];
    array[(int) index + 1] = temp;
    index += 1;
```

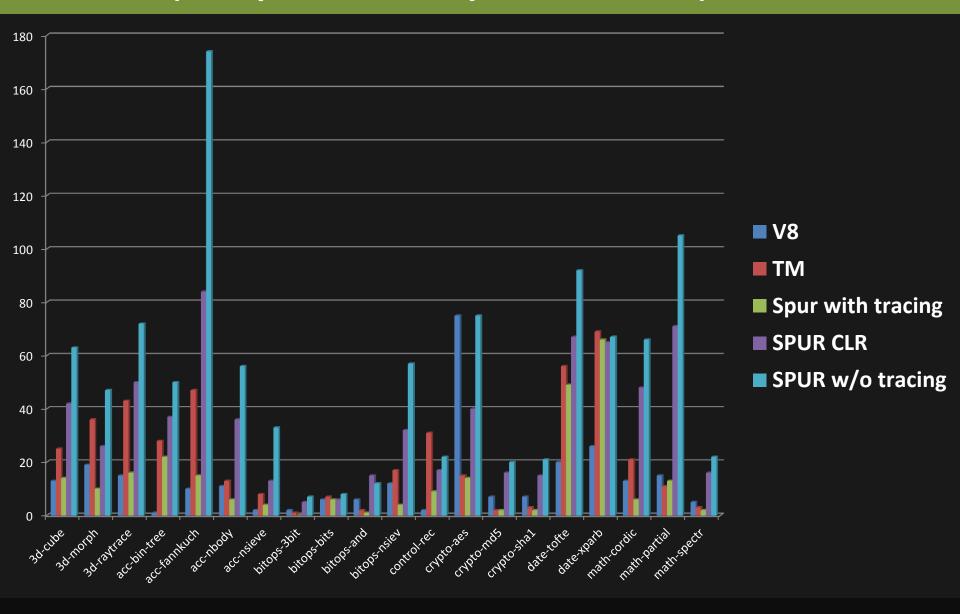
- 41 loop prologue instructions, 18 loop prologue guards
- 10 loop instructions, 2 loop guards!
- From 224 to 10, performance is comparable the C# version of the loop, 7x faster than the CLR operating on JScript code, and slightly faster than V8

# **Optimizing Trace Compiler**

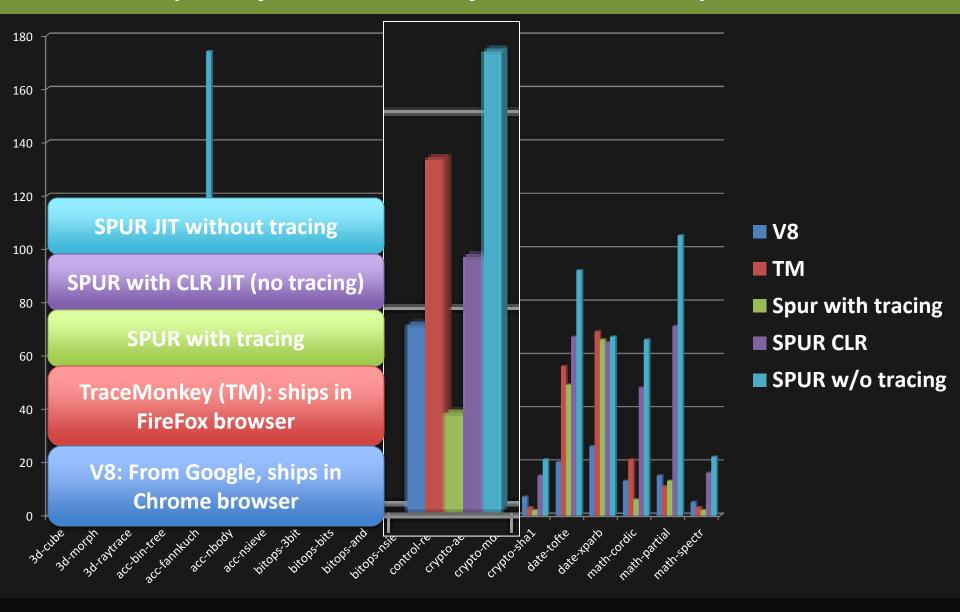
# Standard and not-so-standard optimizations:

- Inlining (for free via tracing)
- Loop unfolding
- Dead Code Eliminations
- Invariant Code Motion
- Constant Folding
- Expression simplification
- Common Subexpression Elimination
- Alias Analysis
- Redundant Guard Elimination
- Redundant Load/Store Elimination
- Speculative Guard Strengthening

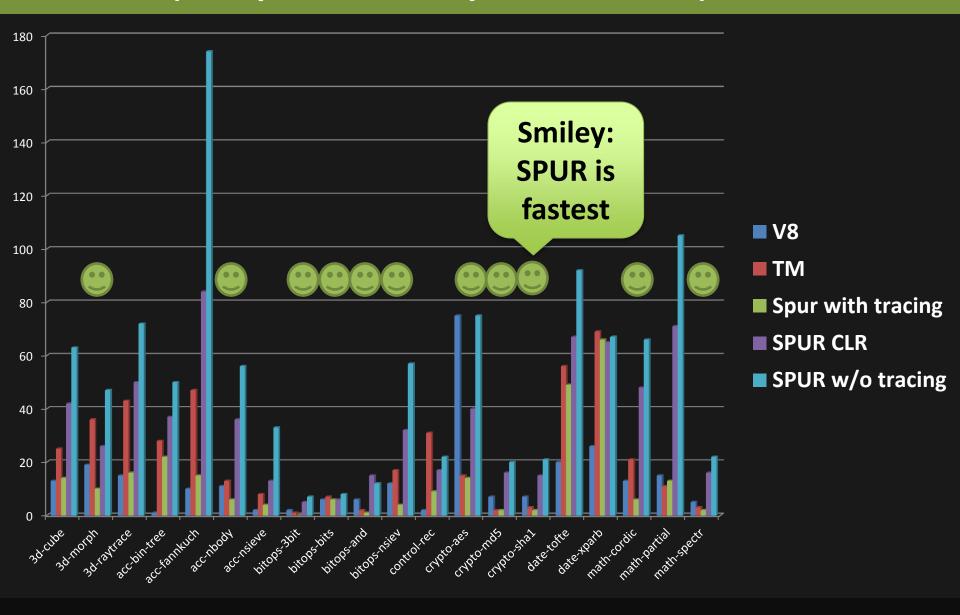
# Numbers (SunSpider JavaScript benchmarks)



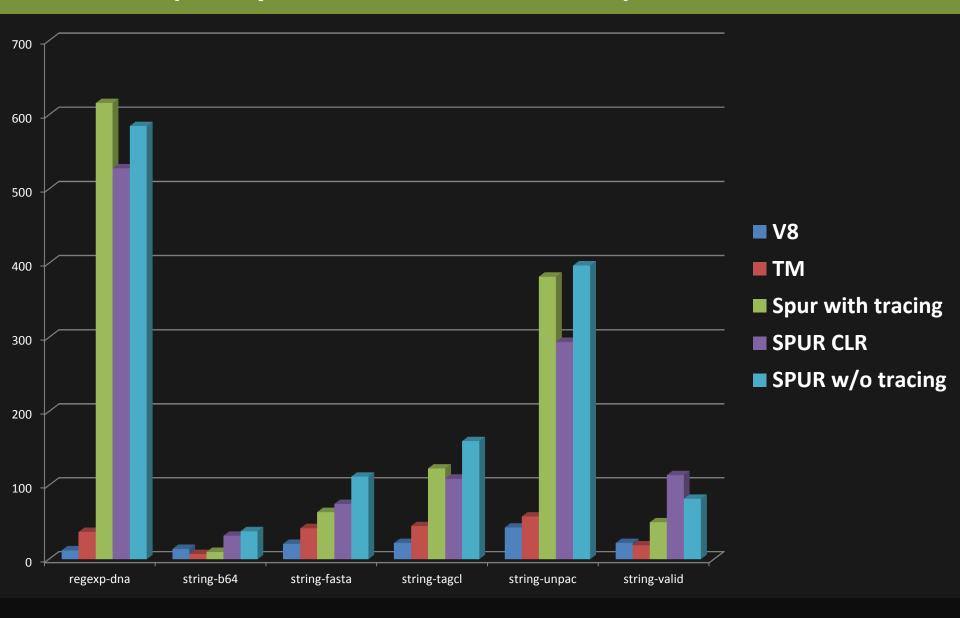
# Numbers (SunSpider JavaScript benchmarks)



# Numbers (SunSpider JavaScript benchmarks)



# Numbers (SunSpider benchmarks cont.)



# **SPUR**

A WALKTHROUGH: JAVASCRIPT

# **Basic idea: Trace Compilation**

```
var sum = 0
for (var i = 0; i < 1000; i++) {
    if (i == 990) {
        sum += " Hello World "
    }
    sum += 1
}
print(sum)</pre>
```

• How would this read in C#?

#### C# equivalent code:

```
enum ValueKind { Double, Object, ... }
struct Value {
 ValueKind Kind;
  double DoubleValue;
  object ObjectValue;
void Main() {
  Value sum;
  SetToDouble(ref sum, 0.0D);
  for (int i = 0; i < 1000; i++) {
    if (i == 990) {
      AddObject(ref sum, "Hello World");
   AddDouble(ref sum, 1.0D);
  Console.WriteLine(ToString(sum));
```

```
// initial state in trace recording:
// sum.Kind == Double
// sum.DoubleValue == 3
i == 3
begin:
  guard i < 1000
  guard i != 990
  begin AddDouble
    guard sum.Kind == Double
    t1 := sum.DoubleValue + 1.0D
    t2 := Value { Kind = Double;
                  DoubleValue = t1; }
  end AddDouble
  t3 = i + 1
  // update state, loop back
  sum := t2, i := t3
  goto begin
```

- SSA form
- At IL level

#### C# equivalent code:

```
enum ValueKind { Double, Object, ... }
struct Value {
  ValueKind Kind
                     When a guard fails,
  double Double\
                     we jump back to
  object Object\
                     unoptimized code

    When a guard fails

                     often, another trace
void Main() {
                     would be recorded
  Value sum;
  SetToDouble(ref sum, 0.0D);
  for (int i = 0; i < 1000; i++) {
    if (i == 990) {
      AddObject(ref sum, "Hello World");
    AddDouble(ref sum, 1.0D);
  Console.WriteLine(ToString(sum));
```

```
// initial state in trace recording:
// sum.Kind == Double
// sum.DoubleValue == 3
// i == 3
begin:
  guard i < 1000
  guard i != 990
  begin AddDouble
   guard sum.Kind == Double
    t1 := sum.DoubleValue + 1.0D
    t2 := Value { Kind = Double;
                  DoubleValue = t1; }
  end AddDouble
  t3 = i + 1
  // update state, loop back
  sum := t2, i := t3
  goto begin
```

- SSA form
- At IL level

#### C# equivalent code:

```
enum ValueKind { Double, Object, ... }
struct Value {
  ValueKind Kind;
  double DoubleValue;
  object ObjectValue;
                   Guard strengthening
void Main() {
  Value sum;
  SetToDouble(ref sum, 0.0D);
  for (int i = 0; i < 1000; i++) {
    if (i == 990) {
      AddObject(ref sum, "Hello World");
    AddDouble(ref sum, 1.0D);
  Console.WriteLine(ToString(sum));
```

```
// initial state in trace recording:
// sum.Kind == Double
// sum.DoubleValue == 3
// i == 3
begin:
 guard i < 1000
 guard i != 990
  begin AddDouble
    guard sum.Kind == Double
   t1 := sum.DoubleValue + 1.0D
   t2 := Value { Kind = Double;
                  DoubleValue = t1; }
  end AddDouble
  t3 = i + 1
  // update state, loop back
  sum := t2, i := t3
  goto begin
```

- SSA form
- At IL level

#### C# equivalent code:

```
enum ValueKind { Double, Object, ... }
struct Value {
 ValueKind Kind;
  double DoubleValue;
  object ObjectValue;
                   Inlining
void Main() {
  Value sum;
  SetToDouble(ref sum, 0.0D);
  for (int i = 0; i < 1000; i++) {
    if (i == 990) {
      AddObject(ref sum, "Hello World");
    AddDouble(ref sum, 1.0D);
  Console.WriteLine(ToString(sum));
```

```
// initial state in trace recording:
// sum.Kind == Double
// sum.DoubleValue == 3
i == 3
begin:
  guard i < 1000
  guard i != 990
  begin AddDouble
    guard sum.Kind == Double
    t1 := sum.DoubleValue + 1.0D
    t2 := Value { Kind = Double;
                 DoubleValue = t1; }
 end AddDouble
  t3 = i + 1
  // update state, loop back
  sum := t2, i := t3
  goto begin
```

- SSA form
- At IL level

#### C# equivalent code:

```
enum ValueKind { Double, Object, ... }
struct Value {
 ValueKind Kind;
  double DoubleValue;
  object ObjectValue;
                   Invariant Code Motion
void Main() {
  Value sum;
  SetToDouble(ref sum, 0.0D);
  for (int i = 0; i < 1000; i++) {
    if (i == 990) {
      AddObject(ref sum, "Hello World");
    AddDouble(ref sum, 1.0D);
  Console.WriteLine(ToString(sum));
```

```
// initial state in trace recording:
// sum.Kind == Double
// sum.DoubleValue == 3
i == 3
begin:
  guard i < 1000
  guard i != 990
  begin AddDouble
    guard sum.Kind == Double
    t1 := sum.DoubleValue + 1.0D
    t2 := Value { Kind = Double;
                  DoubleValue = t1; }
  end AddDouble
  t3 = i + 1
  // update state, loop back
  sum := t2, i := t3
  goto begin
```

- SSA form
- At IL level

#### C# equivalent code:

```
enum ValueKind { Double, Object, ... }
struct Value {
  ValueKind Kind;
  double DoubleValue;
  object ObjectValue;
                          Dead-code
void Main() {
                          elimination
  Value sum;
  SetToDouble(ref sum, 0.0D);
  for (int i = 0; i < 1000; i++) {
    if (i == 990) {
      AddObject(ref sum, "Hello World");
    AddDouble(ref sum, 1.0D);
  Console.WriteLine(ToString(sum));
```

```
// initial state in trace recording:
// sum.Kind == Double
// sum.DoubleValue == 3
i == 3
begin:
  guard i < 1000
  guard i != 990
  begin AddDouble
    guard sum.Kind == Double
    t1 := sum.DoubleValue + 1.0D
    t2 := Value { Kind = Double;
                  DoubleValue = t1; }
  end AddDouble
  t3 = i + 1
  // update state, loop back
  sum := t2 i := t3
  goto begin
```

- SSA form
- At IL level

#### C# equivalent code:

```
enum ValueKind { Double, Object, ... }
struct Value {
  ValueKind Kind;
  double DoubleValue;
  object ObjectValue;
void Main() {
  Value sum;
  SetToDouble(ref sum, 0.0D);
  for (int i = 0; i < 1000; i++) {
    if (i == 990) {
      AddObject(ref sum, "Hello World");
   AddDouble(ref sum, 1.0D);
  Console.WriteLine(ToString(sum));
```

#### **Optimized trace:**

```
prologue:
    guard i < 990
    guard sum.Kind == Double

loopBegin:
    guard i != 990
    t1 := sum.DoubleValue + 1.0D
    // update state, loop back
    i = i + 1
    (&sum)->DoubleValue = t1
    goto loopBegin
```

# **SPUR**

ANOTHER EXAMPLE: C# (NO JAVASCRIPT)

# Fluffy C#

```
class Container {
  private Data data;
  public virtual Data Data {
   get {
      if (data == null) data = new Data();
      return data;
class Data {
  private int[] elements = new int[1000];
  public virtual int Count {
    get { return elements.Length; } }
  public virtual int GetElement(int index)
    if (index < 0 || index > elements.Length)
      throw new ArgumentOutOfRangeException();
    return elements[index];
```

```
int Find(Container c,
         int element)
  for (int i = 0;
       i < c.Data.Count;</pre>
       i++)
    if (c.Data.GetElement(i) ==
        element)
      return i;
  return -1;
```

# Fluffy C#

```
class Container {
  private Data data;
 public virtual Data Data {
   get
      if (data == null) data = new Data();
      return data;
class Data {
  private int[] elements = new int[1000];
 public virtual int Count {
   get { return elements.Length; } }
  public virtual int GetElement(int index)
   if (index < 0 | index > elements.Length)
      throw new ArgumentOutOfRangeException();
   return elements[index];
```

```
int Find(Container c,
         int element)
  for (int i = 0;
       i < c.Data.Count;</pre>
       i++)
    if (c.Data.GetElement(i) ==
        element)
      return i;
  return -1;
```

Virtual calls, lazy initialization, (repeated) argument validation, ...

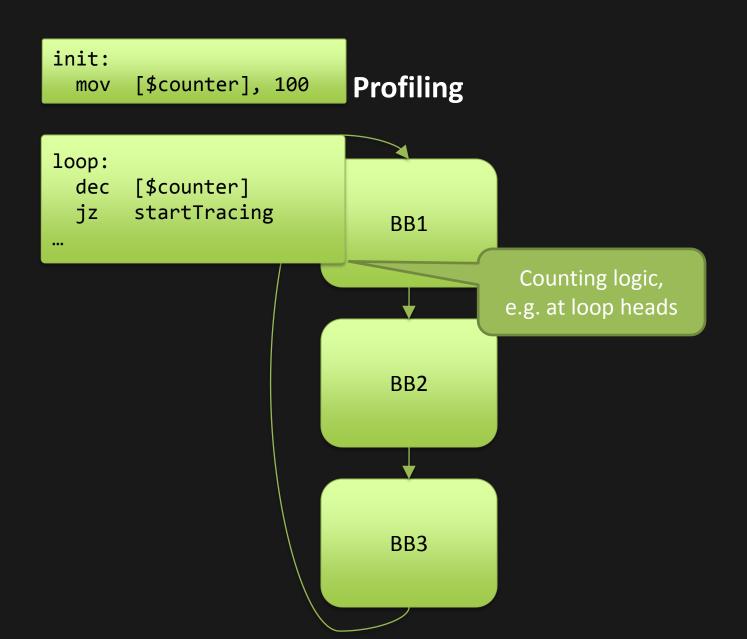
# Fluffy C#: CLR vs. Spur

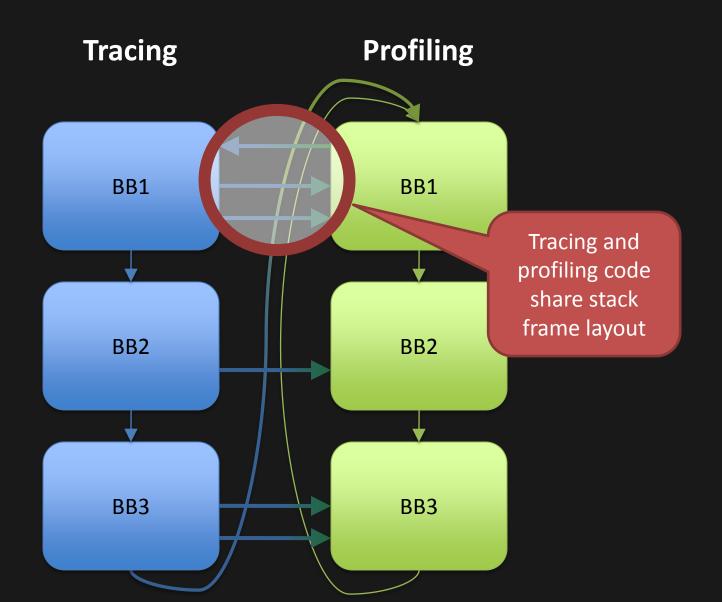
```
class Container {
  private Data data;
  public virtual Data Data {
   get {
      if (data == null) data = new Data();
      return data;
class Data {
  private int[] elements = new int[1000];
 public virtual int Count {
   get { return elements.Length; } }
  public virtual int GetElement(int index)
   if (index < 0 || index > elements.Length)
      throw new ArgumentOutOfRangeException();
    return elements[index];
```

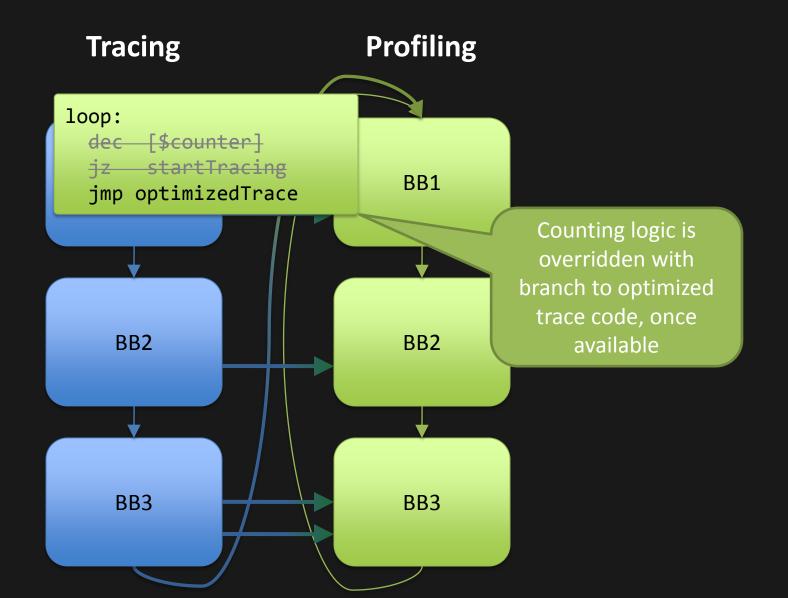
```
int Find(Container c,
         int element)
  for (int i = 0;
       i < c.Data.Count;</pre>
       i++)
    if (c.Data.GetElement(i) ==
        element)
      return i;
  return -1;
```

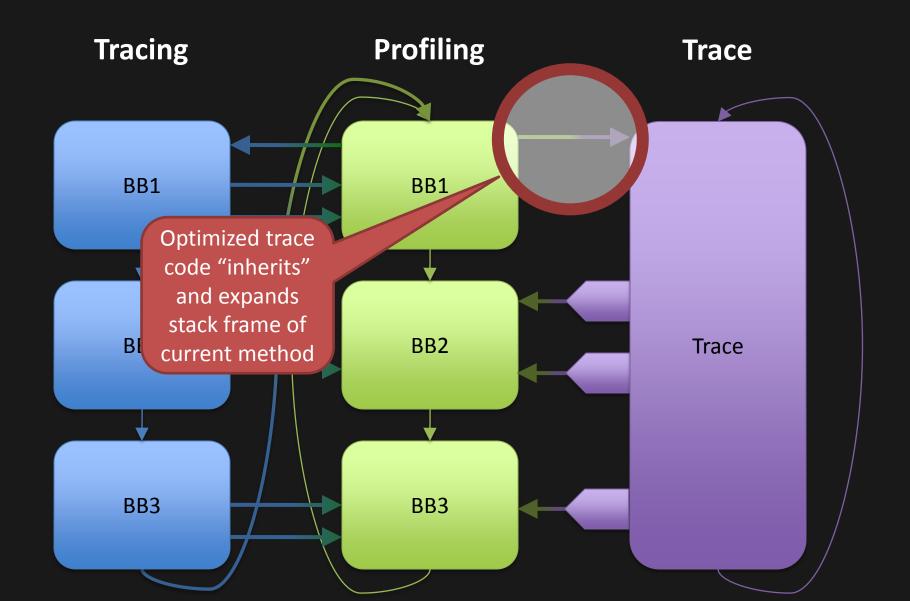
- Running on my laptop (Core 2 Duo, 2.5 Ghz)
- On 32-bit CLR: 1.20s
   With Spur: 0.20s
   Factor of 6

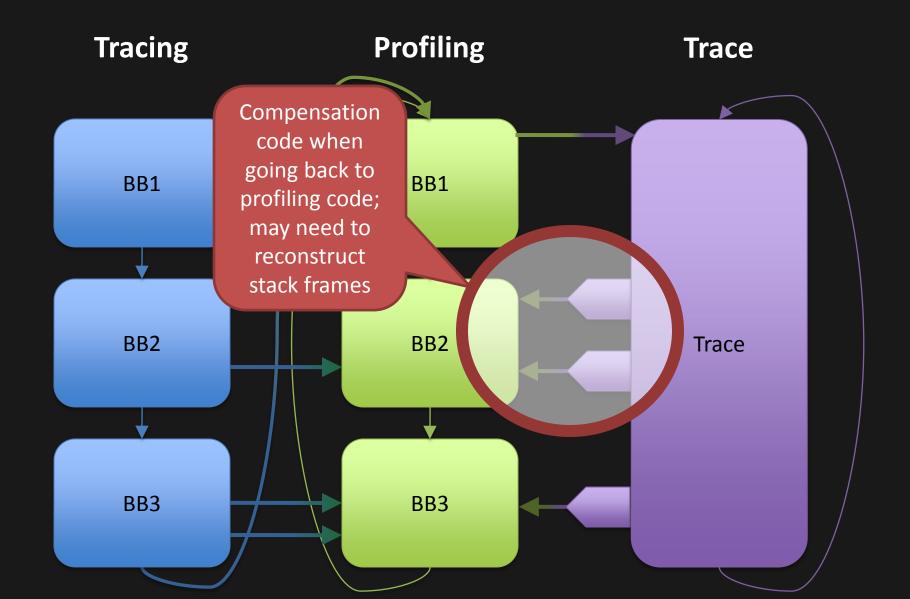
# SPUR UNDER THE HOOD

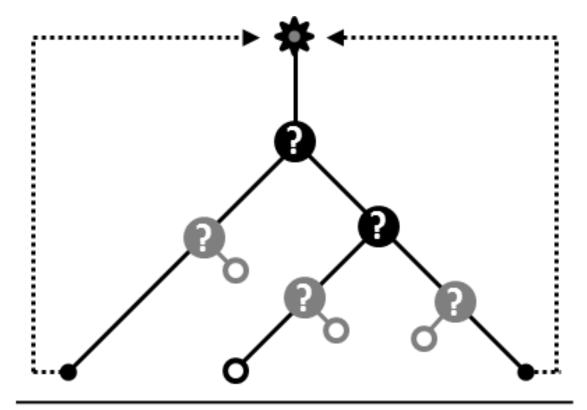


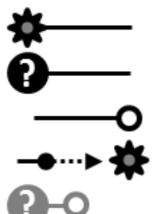












trace anchor

guard with attached trace non-looping trace with exit looping trace guard with trace exit

# **SPUR**

THE FUTURE

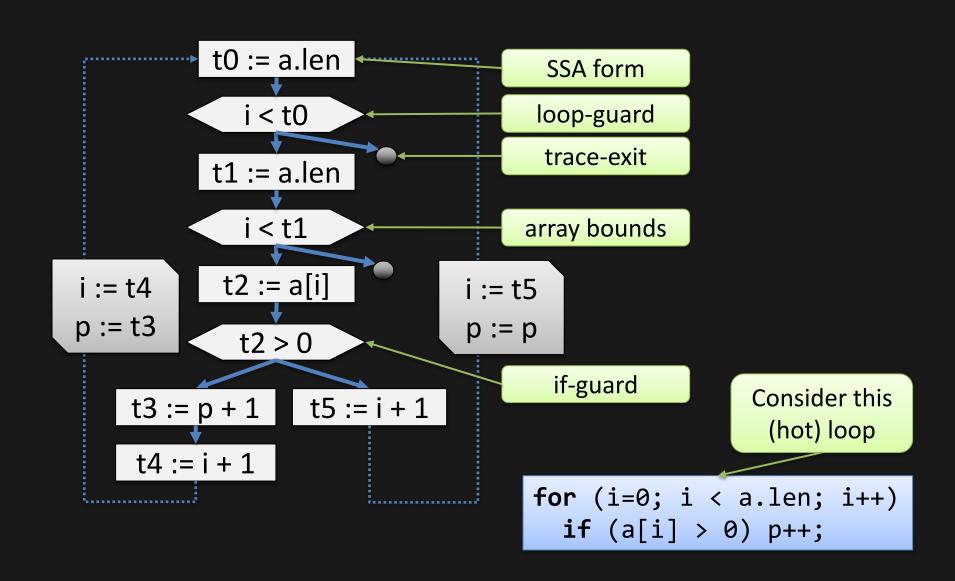
## More Research, Potential Applications

- Advanced trace optimization:
   Using automated theorem proving (SMT solvers)
- Automated parallelization and vectorization: Leveraging runtime information, overcoming JavaScipt's single-threadedness
- Tracing of C# + JavaScript at the same time: Blurring the boundaries of Browser Framework and Browser Apps

### More Research, Potential Applications

- Advanced trace optimization:
   Using automated theorem proving (SMT solvers)
- Automated parallelization and vectorization: Leveraging runtime information, overcoming JavaScipt's single-threadedness
- Tracing of C# + JavaScript at the same time: Blurring the boundaries of Browser Framework and Browser Apps

#### **Trace Tree Example**



#### **Optimizing Trace Compiler**

### Standard and not-so-standard optimizations:

- Inlining (for free via tracing)
- Loop unfolding
- Dead Code Eliminations
- Invariant Code Motion
- Constant Folding
- Expression simplification
- Common Subexpression Elimination
- Alias Analysis
- Redundant Guard Elimination
- Redundant Load/Store Elimination
- Speculative Guard Strengthening

Based on pattern matching;

More patterns → more optimizations

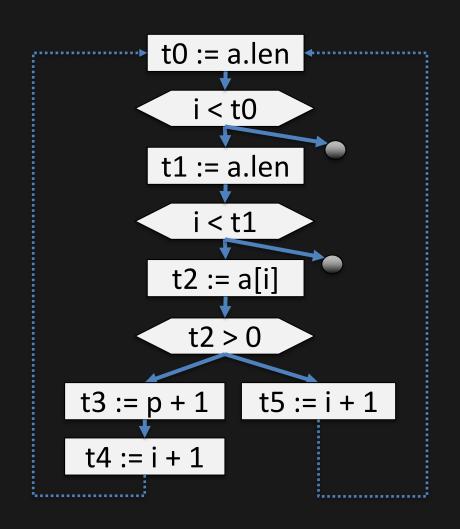
More patterns → more bugs...

#### **Trace Tree Optimizations via SMT Solver**

- Correctness burden: compiler -> SMT solver
- Profiling/Tracing: what's worth optimizing
- Enables deep semantic transformations

- Forward Guard Elimination
- Redundant Store Elimination
- Common Subexpression Elimination
  - modulo theories and asserted guards, including aliasanalysis and redundant-load elimination.
- Speculative Guard Strengthening

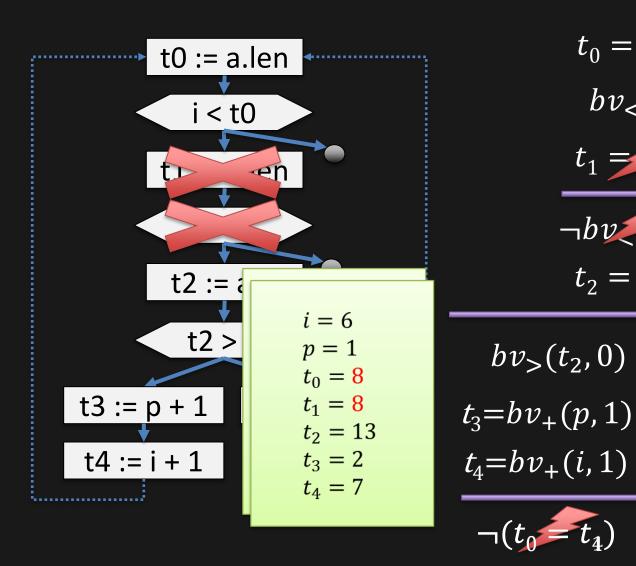
#### **Trace Tree Example**



Consider this (hot) loop

```
for (i=0; i < a.len; i++)
  if (a[i] > 0) p++;
```

### **Trace Tree Optimization Example**



$$t_0 = len_0[a]$$
 $bv_{<}(i, t_0)$ 
 $t_1 = cn_0[a]$ 
 $\neg bv_{<}(i, t_1)$ 
 $t_2 = a[i]$ 
 $bv_{>}(t_2, 0)$   $\neg bv_{>}(t_2, 0)$ 

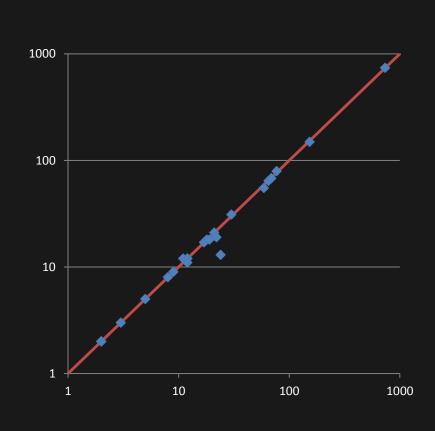
 $t_3 = bv_+(p, 1)$ 

# **C# benchmarks**

		additional
	change in	instructions
program	running time	removed
Alloc2	-0.8%	3.95%
Cmp	-0.8%	0.00%
Grep	-4.1%	0.36%
Linpack	1.0%	6.43%
Pi	0.0%	7.22%
Sat_solve	-2.2%	2.14%
SciMark	0.6%	7.01%
Sieve	-14.1%	6.46%
Sort	0.5%	0.45%
Wc	-2.0%	0.51%

# **JavaScript SunSpider benchmarks**

		additional
	improvement	instructions
program	in running time	removed
3d-morph	9%	2.2%
access-binary-trees	3%	1.8%
access-fannkuch	0%	0.6%
access-nsieve	0%	0.5%
bitops-3bit-bits-in-byte	0%	0.0%
bitops-bits-in-byte	0%	0.0%
bitops-bitwise-and	-46%	2.2%
bitops-nsieve-bits	0%	0.5%
controlflow-recursive	0%	0.1%
crypto-sha1	0%	0.3%
math-cordic	0%	3.9%
math-partial-sums	0%	0.8%
math-spectral-norm	0%	0.3%
regexp-dna	0%	0.7%
string-base64	-8%	4.2%
string-fasta	3%	0.8%
string-tagcloud	-3%	1.0%
string-validate-input	-2%	0.3%
3d-cube	-14%	1.0%
access-nbody	0	1.5%
crypto-aes	-5%	1.6%
crypto-md5	0	0.1%
date-format-tofte	-7%	0.4%
date-format-xparb	-1%	1.2%



#### **Automatic Parallelization: Two challenges**

#### We have to know:

1) Is it worthwhile to parallelize?

Needed: Actual workload, i.e. quantitative data

2) Is it safe to parallelize?

Needed: Dependency analysis, i.e. read/write., write/write conflicts

#### **Loop Parallelization: Quantified analysis**

After having compiled the "optimized" code, analyze if trace parallelization pays off, i.e. if

$$(I*W_e)/P + I*W_i < I*W$$

#### where

- P parallel threads available
- I number of iterations remaining
- W work per iteration on that trace
- $W \approx (W_e + W_i)$

**Parallelization: Splitting** 

# Split loop into 2 phases

# Traditional approach: Inspector + Executor

- Inspector checks for dependencies / creates schedule
- Executor performs computations, and mutates memory

# Spur also has 2 Phases, but

- Inspector also checks paths conditions of known traces
- If no known trace applies to an iteration, finish parallelization right there, and later record new trace

#### Parallelization: Example

#### From Olden benchmarks: Em3d

```
public void computeNewValue() {
    for(int i = 0; i < this.fromCount; i++)
        this.value -= this.coeffs[i] * this.fromNodes[i].value;
}

public void compute() {
    for(int i = this.eNodes.Count - 1; i >= 0; --i)
        this.eNodes[i].computeNewValue();

    for(int i = this.hNodes.Count - 1; i >= 0; --i)
        this.hNodes[i].computeNewValue();
}
```

### **Preliminary experiments indicate:**

- Splitting into Inspector + Executor: 10-20% overhead
- Near linear speed-up by parallelization

#### **Vectorization**

Vectorization is difficult in general JIT setting, as data must be compact and aligned

But applies well to certain library functions:

System.String

**Equals** 

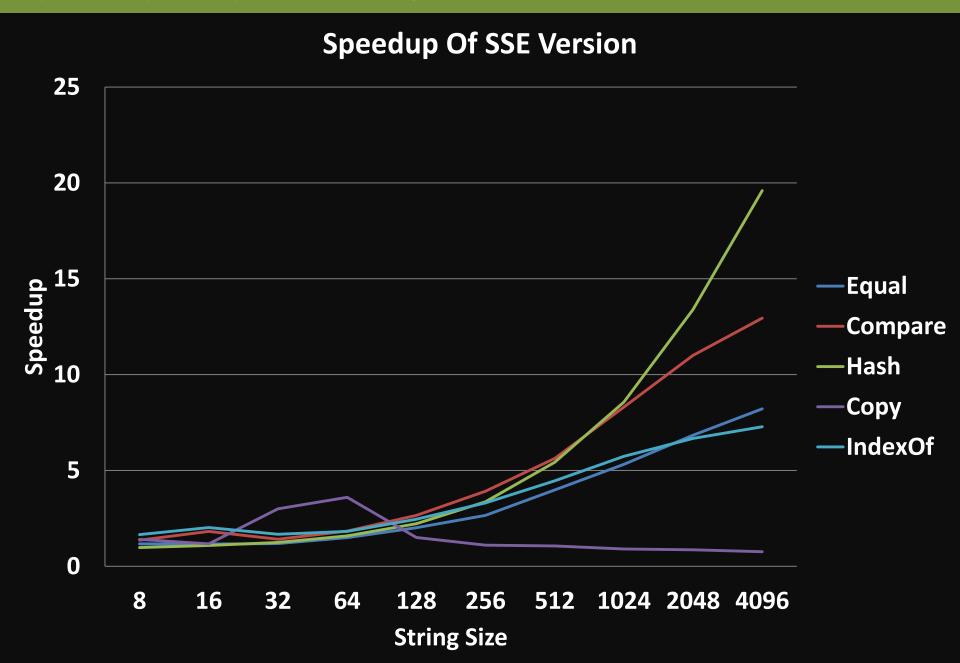
GetHashCode

CompareTo

Copy

IndexOf/IndexOfAny

## **Speed up of System.String functions**



#### **Conclusion and Future Work**

- One runtime + JIT for all languages on .NET!
- Yields excellent runtime performance
- Research platform
- We are implementing and evaluating
  - Automatic parallelization and vectorization
  - Leveraging SMT solver
  - Blurring boundaries: browser framework vs. apps

http://research.microsoft.com/Spur