Compiling X10 to Java

Mikio Takeuchi, Yuki Makino†, Kiyokuni Kawachiya, Hiroshi Horii, Toyotaro Suzumura, Toshio Saganuma, and Tamiya Onodera

IBM Research – Tokyo †IBM Yamato Software Development Laboratory
Outline

- Value Proposition of Managed X10

- Challenges in Compiling X10 to Java
  - Types
  - Generics
  - Arrays

- Experimental Results
  - Sequential Benchmarks
  - Parallel Benchmarks
  - Distributed Benchmarks

- Conclusions

- Future Work
“Managed X10” (X10 on Java VMs)

Managed X10 is an implementation of X10 on Java
- Is a tool for Java programmers to easily scale-out their existing applications with built-in distributed execution feature and better inter-operability with Java.

```
class MyHello {
  public static def main(Array[String]) {
    for (pl in Place.places()) {
      async at (pl) Console.OUT.println(
        "Hello World from place " + here.id);
    }
  }
}
```

Hello World in X10

```
$ x10c MyHello.x10 # Compile
$ x10 MyHello # Run
Hello World from place 3
Hello World from place 0
Hello World from place 1
Hello World from place 2
```

Managed X10

X10 program is compiled into Java source, and executed on Java VMs
**X10 as a co-language for Java**

- **In Java**...
  - How do you deal with petabytes of data?
  - How do you take advantage of GPUs and FPGAs?

- **In X10** – Performance and Productivity at Scale

X10 features
- Places
- Asyncs / Finish / Atomic
- Clocks
- ...

Managed X10 is one of key components in this architecture
A Sample X10 Program

Java-like syntax, but uses var, val, def, ...

Operators can also be defined

New 1st class data types, structs and functions

Strong type system – type parameters are not erased

Primitive types (Int, Double, ...) are defined as structs

Rich array class

Parallel/distributed processing

Global data access
Challenges in Compiling X10 to Java

- **Types** – mapping of X10 data types to Java
  - How to translate X10 types (incl. structs and functions) to Java?
  - How to utilize Java primitive types?

- **Generics** – gaps between X10 generics and Java generics
  - How to implement X10’s richer generics semantics?

- **Arrays** – optimizations of array access
  - How to make X10 Array performance comparable to Java?

- **Places** – distributed execution
  - How to utilize multiple computer nodes to run an X10 application?

- and others ...
X10 Types

- X10 has richer types than Java, and they need to be mapped to Java

  ```
  interface x10.lang.Any
  class x10.lang.Object
  struct x10.lang.Int
  struct x10.lang.UInt
  struct MyStruct
  interface x10.util.Map[K,V]
  interface MyInterface
  ```

- Interface “Any” as a top-level
- Structs and functions
- Int (and UInt, ...) are structs
- No “int” or built-in array
Mapping X10 Types to Java (in X10 2.2.0)

- Interfaces to represent class/struct/function
- Some X10 types are mapped to Java primitives or well-known Java classes
- Throwable are under RuntimeException

Unsigned types will be mapped to Java primitives in X10 2.2.1
Example of Compilation

Class/field names are basically same

Constructor is separated into Java constructor and field initializer

For private instance method, additional bridge method is generated
  – to support inlining by front-end

Static-field initializer and deserializer are generated
  – to support multi-place initialization
Static Initialization

- In X10, static fields are immutable and have the same value in all places.
- General approach: evaluate static fields in place 0 and broadcast them to all other places.
  - Difficulty of the use of final keyword impacts performance.
- Optimization: evaluate as many static fields as possible in each place if compiler can determine it is safe.

X10 also requires all static fields must be initialized successfully before the main method is executed.

Current approach: preload all X10 classes.

- Large footprint
- Massive class loading in a short time has impact to some JIT compilers (e.g. IBM J9)
Generics

- X10 generics must be implemented by **type reification**
  - Type parameters are kept for each instance, like in C++ templates
    - e.g. “new Sample[Int](1).typeName()” returns “Sample[x10.lang.Int]”

- However, Java generics are implemented by **type erasure**
  - Type parameters are checked and erased by javac

Quiz: How this X10 code can be translated to Java?

```java
1 interface I[T] {
2   def m(T):T;
3 }
4 class B[T] {
5   def m(a:T):T {return a;}
6 }
7 class C[T1,T2] extends B[Int] implements I[String], I[Int] {
8   def this(T1){}
9   def this(T2){}
10  def m(a:T1){return a;}
11  def m(a:T2){return a;}
12  public def m(a:String){return a;}
13  public def m(a:Int){return a;}
14 }
```

- How to implement both I[String] and I[Int]?
- How to overload m(T1) and m(T2)?
- How to pass primitive (unboxed) data to get better performance, while allowing access through B[Int]
Generics Compilation

X10 generics are mapped to Java generics, but ...

- Static field "$RTT" holds X10-level class info
- Dispatch method is generated for overloading
- Method name is modified to include type parameters
- Primitives are used as much as possible, and bridge method is generated to access it through boxed types

Java

```
1 import x10.rtt.RuntimeType;
2 import x10.rtt.Type;
3 import x10.rtt.Types;
4 interface I<T> {
  5   public static final RuntimeType<I> $RTT = ...
  6   Object m(T id$0, Type t1);
  7 }
8 public class B<T> extends x10.core.Ref {
  9   public static final RuntimeType<B> $RTT = ...
10    private Type T;
11    public T m_0_$$B_T$G(T a) {return a;}
12 }
13 public class C<T1, T2> extends B<Integer> implements I<String>, I<Integer> {
14    private Type T1, T2;
15    // dispatcher for abstract public I.m(id$0:T):T
16    public Object m(Object a1, Type t1) {
17      if (t1.equals(Types.STRING)) {
18        return m((String) a1);
19      } else if (t1.equals(Types.INT)) {
20        return m((int)(Integer) a1);
21      } return null;
22    }
23    // bridge for B.m(a:T):T
24    public Integer m_0_$$B_T$G(Integer a1) {return m((int) a1);}
25    // constructors need signature mangling
26    public C(Type T1, Type T2) {
27      T1 id$1, Class dummy0) {...}
28      T2 id$2, Class[] dummy0) {...
29    // generic methods need signature mangling
30    public T1 m_0_$$C_T1$G(T1 a) {return a;}
31    public T2 m_0_$$C_T2$G(T2 a) {return a;}
32    // instantiated generic methods
33    public String m(String a) {return a;}
34    public int m(int a) {return a;}
35    // instantiated generic methods
36    public String m(String a) {return a;}
37    public int m(int a) {return a;}
38 }
```
Arrays

- X10 array is not a built-in data type, but is a class
  - It is generic, multi-dimensional, and sparse array
- An X10 array instance consists of three objects
  - `Array[T]` holds array attributes
  - `IndexedMemoryChunk[T]` represents a contiguous 1-dimensional array
  - Actual Java array (e.g. `String[]` or `int[]`) to hold array elements

To get performance, Array access is (should be) inlined and privatized in some situations

```
1 val arr = new Array[Int](3..5);
2 for (i in 3..5) arr(i) = i
3 var s:Int = 0;
4 for (pt in arr) s += arr(pt)*arr(pt);
5 Console.OUT.println(s); // -> 50
```
Array Access (e.g. Array[Int]{rail})

```
1 public final class Array[T]
2   (region:Region{self!=null}, rank:Int, rail:Boolean) {...}
3 implements (Point(rank))=>T, Iterable[Point(rank)] {
4 private val raw:IndexedMemoryChunk[T],
5 private val layout:RectLayout;
6 public operator this(i0:Int){rank==1}:T {
7     if (rail) {
8       return raw(i0);
9     } else {
10       if (CompilerFlags.checkBounds() && !region.contains(i0)) {
11         raiseBoundsError(i0);
12       }
13       return raw(layout.offset(i0));
14   } }
15 public operator this(i0:Int)=(v:T){rank==1}:T{self==v} {
16     if (rail) {
17       raw(i0) = v;
18     } else {
19       if (CompilerFlags.checkBounds() && !region.contains(i0)) {
20         raiseBoundsError(i0);
21       }
22       raw(layout.offset(i0)) = v;
23     } return v;
24 }
25 }
```
Optimized Array Access

We have implemented operator inlining and privatization for IndexedMemoryChunk but not for Array.
Distributed Execution

- Multi-JVM is supported in X10 2.1.2
  - Each X10 “Place” uses its own Java VM
  - Uses common X10RT (in C++) through JNI

```
“x10” command
```

```
Place P

JVM

x10.runtime.impl.java.Runtime

Run activity at place P
Serialize function into byte array
Send byte array
JNI (Java->C++)
x10rt_send_msg
Communication Physical Layer (e.g. socket, MPI)

Byte array message
Network

Uses backend unique wire format in X10 2.2.0
```

```
Place Q

JVM

x10.runtime.impl.java.Runtime

Invoke function
Deserialize function from byte array
Receive byte array from place P
JNI (Java->C++)
x10rt_probe
Communication Physical Layer (e.g. socket, MPI)

Byte array message
```

```
Initiate Places via SSH
```

---

© 2011 IBM Corporation
IBM Research - Tokyo

16 2011 X10 Workshop (X10’11)
Performance Improvement

- **Performance on single JVM was much improved in 2.1.2**
  - Both in sequential (a), and parallel (b) benchmarks

- **Multi-JVM also shows good scalability (c)**
  - Need further tuning

---

**Intel Xeon X5670 (6-core, SMT-off, 2.93GHz) x 2, 16GB memory**
64-bit Red Hat Enterprise Linux Server release 5.5 (kernel 2.6.18-194.el5)
IBM J9 VM (build 2.4, JRE 1.6.0 IBM J9 2.4 Linux amd64-64
jvmxa6460sr9-20110203_74623 (JIT enabled, AOT enabled)).
Conclusions

- Presented value proposition of Managed X10
- Explained challenges in compiling X10 to Java
  - For performance and functionality
- Discussed compilation techniques
- Demonstrated performance improvement history
Future Work

- Better Java Interoperability
- Heterogeneous Interoperability
- Sequential Performance (Array, Map, etc.)
- Parallel Performance (atomic)
- Distributed Performance and Scalability (Multi-JVM)
- Smaller Footprint
Thank You