IBM

The APGAS Library

Resilient Parallel and Distributed Programming in Java 8

Olivier Tardieu

IBM Research

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research under Award Number DE-SC0008923.

The Equation

```
   the X10 language
 - the     language
-------------------
 =     ???
```

The Equation

```
  the X10 language
- the     language
-------------------
= the APGAS library
```

Take Away

The APGAS library supports

- resilient, elastic, parallel, distributed programming on clusters of JVMs
- for the Java programmer
- as a pure Java library
- for the Scala programmer
- as an embedded DSL in Scala

Compared to X10

- same programming model
  - places, lightweight tasks, finish, global heap
- X10 offers a lot more
  - syntax, type system, C++/CUDA backend, program analysis and optimization
- for a price
  - new language, 200K LOCs implementation

http://x10-lang.org/software/download-apgas/latest-apgas-release.html

Outline

- API and Examples
- Implementation
- Experimental evaluation
- Roadmap
- Demo

HelloWorld

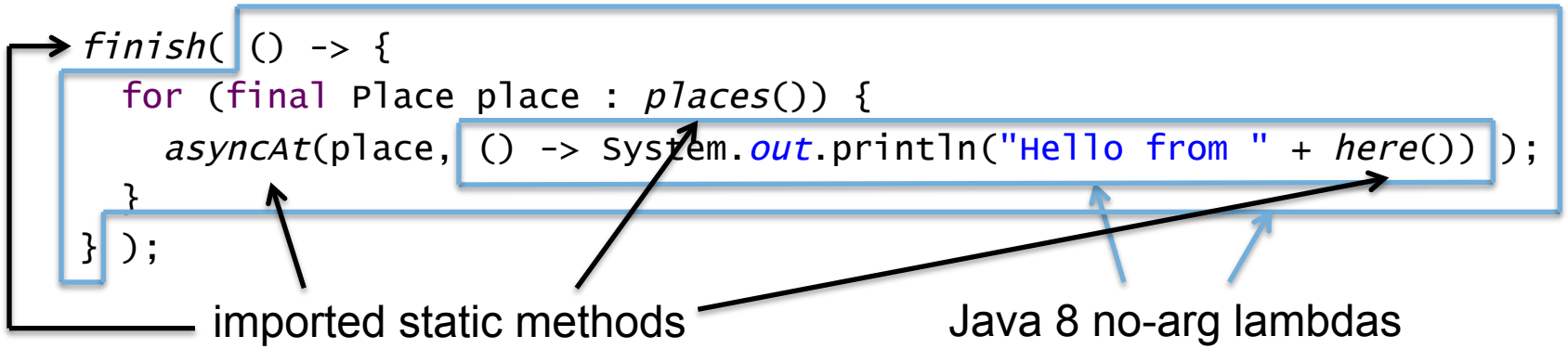
- X10

```
finish for (place in Place.places()) {
  at(place) async Console.OUT.println("Hello from " + here);
}
```

- APGAS

```
import static apgas.Constructs.*;
import apgas.Place;

finish( () -> {
  for (final Place place : places()) {
    asyncAt(place, () -> System.out.println("Hello from " + here()) );
  }
} );
```



Compiling and Running

- Compile
  - `javac -cp .:apgas.jar HelloWorld.java`
- Run with one place
  - `java -cp .:apgas.jar:hazelcast.jar HelloWorld`
- Run with 4 places (single host)
  - set “apgas.places” system property at launch time
    - `java -Dapgas.places=4 -cp .:apgas.jar:hazelcast.jar HelloWorld`
  - set “apgas.places” system property in main
    - `System.setProperty(Configuration.APGAS_PLACES, "4");`
- Configure desired parallelism (typically not required)
  - set “apgas.threads” property

Core API

- class Constructs
  - static void finish(Job f);
  - static void async(Job f);
  - static void asyncAt(Place p, SerializableJob f);
  - static void at(Place p, SerializableJob f); // equivalent to finish async
  - static <T extends Serializable> T at(Place p, SerializableCallable<T> f);
  - static Place here();
  - static Place place(int id);
  - static List<? extends Place> places();

- class Place
  - Place(int id);
  - int id();

- functional interfaces Job, SerializableJob, SerializableCallable<T>

Fibonacci

- X10

```
static def fib(n:long) {
  if (n<2) return n;

  val v0:long; val v1:long;
  finish {
    async { v0 = fib(n-2); }
    v1 = fib(n-1);
  }
  return v0+v1;
}

// static def/use analysis
// compiles to C++ and Java
// stack-allocated locals (C++)
// autoboxing (Java)
```

- APGAS

```
static long fib(final long n) {
  if (n<2) return n;

  final long v[] = new long[2];
  finish(() -> {
    async(() -> v[0] = fib(n-2));
    v[1] = fib(n-1);
  });
  return v[0]+v[1];
}

// no static analysis
// Java only
// heap-allocated locals
// explicit boxing
```

Global Heap (PGAS)

- `GlobalID`
  - globally unique handle
- `GlobalRef<T>`
  - union of X10's `GlobalRef` and `PlaceLocalHandle`
  - explicit distributed collection of objects (zero or one per place)
  - dereferencing a global reference returns the object local to the place (if any)
- `PlaceLocalObject`
  - implicit distributed collection of objects (zero or one per place)
  - "scoped static field"
  - serializing one object in the collection serializes the collection id
  - deserialization resolves the id to the local object
- `PlaceLocalArray<T>`, `PlaceLocalIntArray`…

Example

```
class Foo extends PlaceLocalObject {
  int value;

  public Foo() {
    value = here().id + 42;
  }

  public static void main(String[] args) {
    System.setProperty(Configuration.APGAS_PLACES, "2");

    final Foo foo = Foo.make(places(), () -> new Foo());
    System.err.println(here() + ": " + foo.value);       // place(0): 42
    at(place(1), () -> {
      System.err.println(here() + ": " + foo.value);     // place(1): 43
    });
  }
}
```

Implementation Highlights

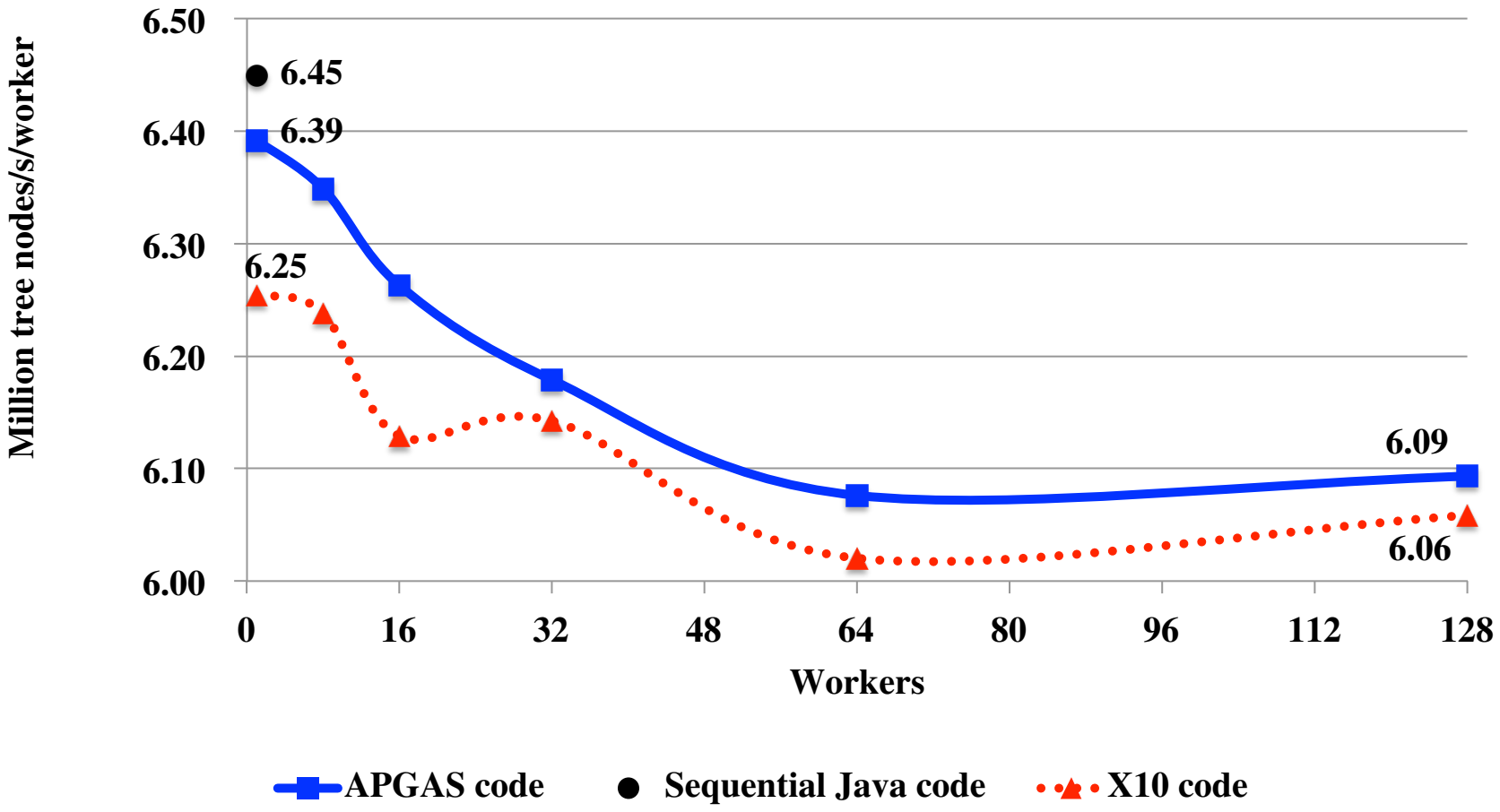
- Java 8 lambdas
  - like X10: immutable environment capture, unlike X10: no autoboxing of vars
- Java fork/join thread pool and scheduler
  - APGAS twist: ad hoc thread creation and disposal policies
- Java serialization
  - unlike X10: must implement java.io.Serializable, but type inference for lambdas
  - APGAS twist: compiler warnings to mitigate runtime serialization exceptions
- Hazelcast
  - fault-tolerant, elastic cluster management
  - in-memory resilient key value store
- Resilient and non-resilient modes

2K LOCS

Unbalanced Tree Search

- Count nodes in dynamically generated tree
  - separable cryptographic random number
- Representative of a class of irregular applications
  - computationally intensive
  - locality insensitive
- APGAS implementation follows from X10
  - lifeline-based global work-stealing
- Configuration
  - 16 servers
    - 2 Quad-Core AMD Opteron(tm) Processor 2356
    - gigabit ethernet network
  - 1 APGAS place per server

Unbalanced Tree Search



Roadmap

- 1.0 release available today
  - library and examples
  - Eclipse integration
- Short term
  - tutorial
  - cloud integration
    - Yarn launcher
  - Scala bindings
- Longer term
  - Spark integration
  - resilient elastic APGAS
  - deeper Eclipse integration

Fibonacci in Scala APGAS

```
def fibonacci(i: Int) : Long = {
  if(i <= 1 ) i else {
    var a,b = 0L
    finish {
      async {
        a = fibonacci(i - 2)
      }
      b = fibonacci(i - 1)
    }
    a + b
  }
}
```

Take Away

The APGAS library supports

- resilient, elastic, parallel, distributed programming on clusters of JVMs
- for the Java programmer
- as a pure Java library
- for the Scala programmer
- as an embedded DSL in Scala

Compared to X10

- same programming model
  - places, lightweight tasks, finish, global heap
- X10 offers a lot more
  - syntax, type system, C++/CUDA backend, program analysis and optimization
- for a price
  - new language, 200K LOCs implementation

http://x10-lang.org/software/download-apgas/latest-apgas-release.html

DEMO