Ajents: Towards an Environment for Parallel, Distributed and Mobile Java Applications

Matthew Izatt, Patrick Chan
York University

Tim Brecht
University of Waterloo

http://www.cs.uwaterloo.ca/~brecht

AJENTS: GOALS

- Simplify writing and running distributed applications (no account required on serving machines)
- Seamless access to heterogenous resources on network
- Parallel, distributed and mobile Java objects
- Serial jobs remotely (faster machines / more memory)

- World-Wide Computing
- Leverage existing Java technology
- **JAVA COMPATIBILITY**

AJENTS: DESIGN

- Java class libraries   (standard and portable)
- Remote object creation   (only need byte-code)
- Synchronous and asynchronous RMI
- Object migration

SUN RMI: REMOTE OBJECT

```
public interface B
  extends java.rmi.Remote
{
  String getName()
    throws java.rmi.RemoteException;
  String setName(String name)
    throws java.rmi.RemoteException;
}

public class BImpl extends UnicastRemote
  implements B
{
  . . .
```

```
prompt> rmic BImpl
```

AJENTS: REMOTE OBJECT

```
public class B
  implements Serializable  // to migrate
{
  String name;
  public B() {
    name = new String("Default");
  }
  public String getName() {
    return name;
  }
  public String setName(String name) {
    String oldname = this.name;
    this.name = name;
    return oldname;
  }
}
```

REMOTE OBJECT CREATION

SUN

- Find available host, install code and run object (`"B1"`)
- At client side ☞ obtain reference to `B1`

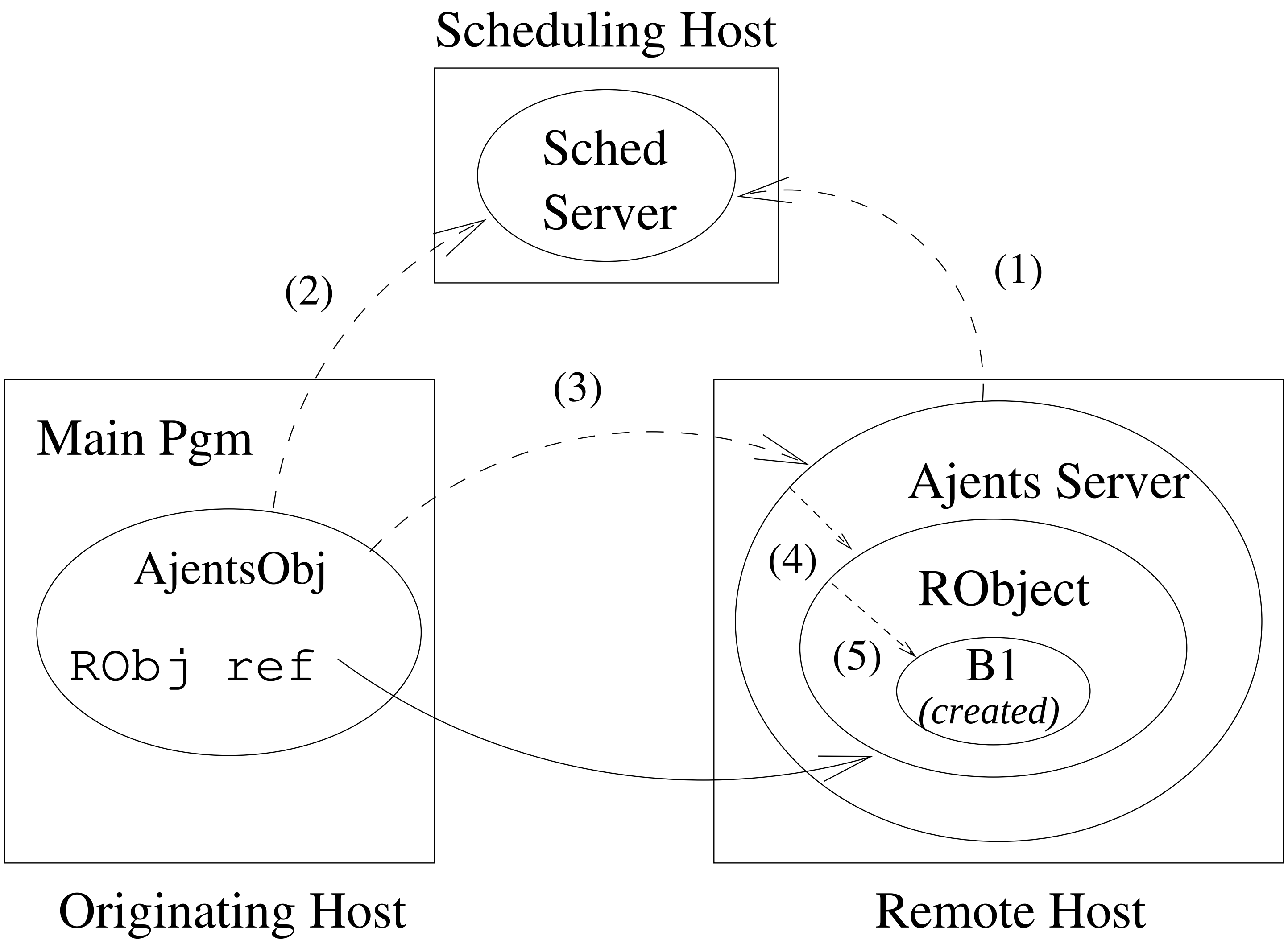
```
B b1 = (B) Naming.lookup("rmi://Host2/B1");
```

AJENTS

- Admins run Ajents server & sched on remote hosts
- At client side ☞ create remote objects

```
AjentsSched s = Ajents.register();
AjentsObj b1 = Ajents.new("ajents.B", "B1",
  "/ajents/myobjs.jar", s.AvailServer());
```

AJENTS: REMOTE OBJECT CREATION



SUN RMI

```
try {
  oldname = b1.setName("NewB1");
  name = b1.getName();
} catch (Exception ex) { ... }
```

AJENTS RMI

```
try {
  oldname = Ajents.rmi(b1, "setName",
    "NewB1");
} catch (Exception ex) { ... }

Future f = Ajents.armi(b1, "getName");

try {
  oldname = (String) f.get();
} catch (Exception ex) { ... }
```

AJENTS: OBJECT MIGRATION

- By the creating object, Ajents server, or 3rd party

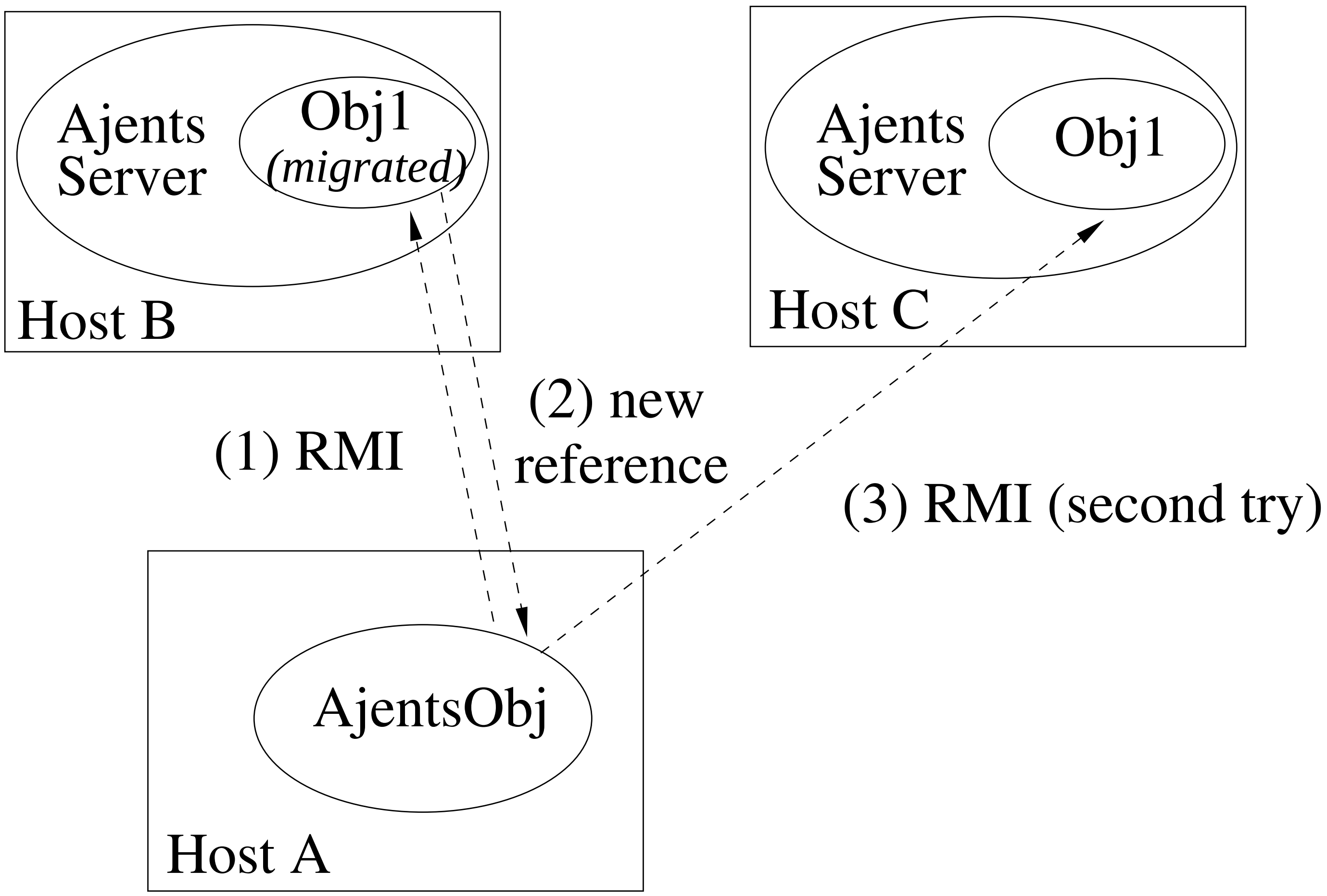
1) Immediate migration of idle objects
2) Delayed migration (until idle)
3) Immediate migration using checkpointing and roll back

```
public int method(int N) {
  for (int i=0; i<N; i++) {
    // do something
    . . .
  }
  return value;
}
```

AJENTS: OBJECT MIGRATION

```
Ajents.migrate(b1, sched.AvailServer());

// Turn checkpointing on
Ajents.setCheckPoint(b1, true);

// b1 checkpointed, method invoked
Future f = Ajents.armi(b1, "getName");

// move b1, possibly before armi finishes
Ajents.migrate(b1, sched.AvailServer());

name = (String) f.get();
```

AJENTS: OBJECT MIGRATION



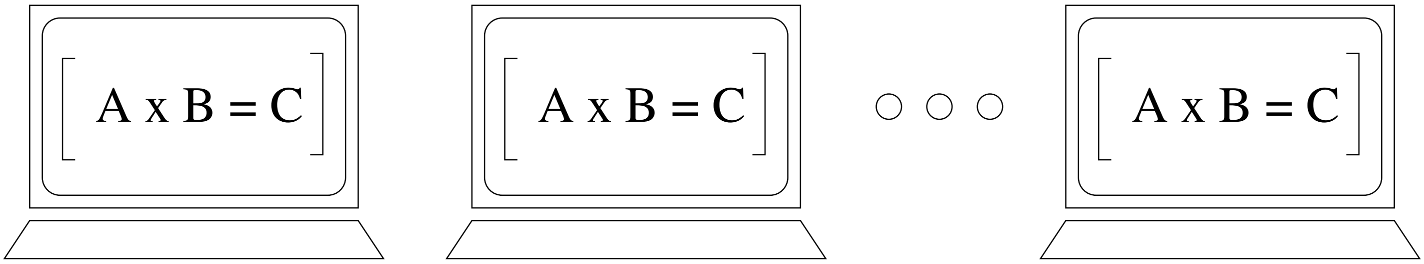
MICRO-BENCHMARK PERFORMANCE

RMI					
Object Size	(ints)	1	1k	10k	100k
RMI - Ajents	(ms)	9	11	46	400
RMI - JDK RMI	(ms)	5	8	44	400

MIGRATION					
Object Size	(ints)	1	1k	10k	100k
Migration time	(ms)	334	334	397	769

CHECKPOINTING					
Object Size	(ints)	1	1k	10k	100k
No-Checkpointing	(ms)	6	6	6	6
Checkpointing	(ms)	8	9	19	115
Checkpointing Cost	(ms)	2	3	13	109

EXAMPLE APPLICATION MIGRATION OVERHEADS



N	200	500	640	800	1024
Migrated	0.5	2.9	4.7	7.3	12.0
1 Server	7.7	134	292	580	1305
8 Servers	20.3	165	344	666	1402
Inflation	2.64	1.23	1.18	1.15	1.07

Times in seconds, migrated data in MB.

Inflation = time on 8 servers / time on 1 server.

PARALLEL APPLICATION PERFORMANCE

N	200	500	640	800	1024
Seq. C	4.0	67	139	280	601
Seq. Java	7.5	131	286	574	1272
1 Server	7.7	134	292	580	1305
8 Servers	1.5	20	40	78	172
Speedup	5.1	6.7	7.3	7.4	7.6

Time in seconds, speedup relative to Seq. Java

RELATED WORK

• ABC++	[Arjomandi *et al.* 95]
• ParaWeb	[Brecht *et al.* 96]
• Mole	[Straßer *et al.* 96]
• SuperWeb	[Alexandrov *et al.* 97]
• Javelin	[Christiansen *et al.* 97]
• JavaParty	[Philippsen & Zenger 97]
• Asynchronous RMI	[Raje *et al.* 97]
• Aglets	[Lange & Oshima 98]
• Fast RMI	[Krishnaswamy *et al.* 98] [Maassen *et al.* 99]

- Many others

CONCLUSIONS

- Java Compatible (pros and cons)
- Remote object creation (only need byte-code)
- Synchronous or asynchronous remote method invocation
- Object migration

- Quite reasonable performance
- Simple to build applications with good performance

`http://www.cs.uwaterloo.ca/~brecht`