Parallel programming in Java
Parallel programming in Java

- Java has 2 forms of support for parallel programming:
  - Multithreading
    - Multiple threads of control (sub processes), useful for
      - Pseudo-parallelism within a single machine
      - Real parallelism on shared-memory machine
  - Remote Method Invocation (RMI)
    - Allows invocation on an object located at another machine
    - Useful for distributed-memory machines
- Many additional parallel programming libraries exist:
  - MPJ Express (based on MPI)
  - Lab course uses IPL (Ibis Portability Layer)
Multithreading

A thread has
- Its own program counter
- Its own local variables

All threads on same Java Virtual Machine share global variables

Threads can communicate through shared variables

Threads can run concurrently (on multiprocessor or multicore) or are time-sliced
Creating threads in Java

public class mythread extends Thread {
    public void hi() {
        System.out.println("hi");
    }
    public void run() {
        System.out.println("hello");
    }
}

mythread t1 = new mythread(); // allocates a thread
mythread t2 = new mythread(); // allocates another thread

 t1.start(); // starts first thread and invokes t1.run()
t2.start(); // starts second thread and invokes t2.run()
t1.hi();
Thread synchronization

Problem-1:
Thread-1 does: $X = X + 1$;
Thread-2 does: $X = X + 2$;
Result should be +3, not +1 or +2.

Need to prevent concurrent access to same data:
mutual exclusion synchronization
Mutual exclusion in Java

```java
public class example {
    int value;

    public synchronized increment (int amount) {
        value = value + amount;
    }
}
```

The synchronized keyword guarantees that only one call to increment is executed at a time
More thread synchronization

Problem-2:
Sometimes threads have to wait for each other
Condition synchronization
Supported in Java with wait/notify/notifyAll

wait blocks (suspends) a thread
Notify wakes up (resumes) one blocked thread
notifyAll wakes up all blocked threads
Remote Method Invocation

RMI is 2-way synchronous communication, like RPC

RMI invokes a method on a (possibly) remote object

Integrates cleanly into Java's object-oriented model
Example

```java
public interface Hello extends Remote {
    String sayHello();
}

public class HelloImpl extends UnicastRemoteObject implements Hello {

    public String sayHello() {
        return "Hello World!";
    }
}
```
Asynchronous communication with Java RMI

Can you do asynchronous messages passing in Java?

Yes: create a new thread to do the RMI for you and wait for the result

Awkward to program, performance overhead
IPL (Ibis Portability Layer)

- Ibis: Java-centric communication system designed for grid computing
  - Supports heterogeneous and dynamic (malleable) systems
  - Discussed in Cluster & Grid class
- IPL: flexible message passing library for Java
Structure of the Ibis system

Legend:  
Java  
Native
Functionality of IPL

• Based on setting up connections
  – Programmer can create send-ports and receive-ports
  – These can be connected in a flexible way: one-to-one, one-to-many (multicast), many-to-one

• Programmer can define properties of connections and ports:
  – FIFO ordering, reliability, delivery mechanisms, streaming

• IPL supports explicit message receipt, implicit message receipt (upcalls), polling
More information

Tutorials/documentation about multithreading and IPL will be made available through the Blackboard site of the lab course
Ibis for Jungle Computing (Frank Seinstra)

- ‘Worst case’ computing as required by end-users
  - Distributed
  - Heterogeneous
  - Hierarchical (incl. multi-/many-cores)
Multimedia Content Analysis (MMCA)

- **Aim:**
  - Automatic extraction of ‘semantic concepts’ from image sets and video streams

- Depending on specific problem & size of data set:
  - May take hours, days, weeks, months, years…
Multimedia Content Analysis (MMCA)

- Applications in (a.o.):
  - Remote Sensing
  - Security / Surveillance
  - Medical Imaging
  - Document Analysis
  - Multimedia Systems
  - Astronomy

- Application types:
  - Real-time vs. off-line
  - Fine-grained vs. coarse-grained
  - Data-intensive / compute-intensive / information-intensive
Color-based Object Recognition by a Grid-connected Robot Dog

Seinstra et al (AAAI'07: Most Visionary Research Award)
Other Application Domains

- Computational Astrophysics (Leiden)
  - AMUSE: multi-model / multi-kernel simulations

- Climate Modeling (Utrecht)
  - CPL: multi-model / multi-kernel simulations
    - Atmosphere, ocean, source rock formation, …
      - hardware: (potentially) very diverse
      - high resolution => speed & …
Computational Astrophysics

- The AMUSE system (Leiden University)
  - Early Star Cluster Evolution, including gas

- Gravitational dynamics (N-body): GPU / GPU-cluster
- Stellar evolution: Beowulf cluster / Cloud
- Hydro-dynamics, Radiative transport: Supercomputer
Ibis Results: Awards & Prizes

1st Prize: SCALE 2008 - BS
1st Prize: DACH 2008 - FT
Euro-Par Achievement Award 2014

AAAIC-VC 2007
Most Visionary Research Award

3rd Prize: ISWC 2008
1st Prize: SCALE 2008
1st Prize: SCALE 2010
EYR 2011 Sustainability award

- Many domains; data/compute intensive, real-time...