Note on the proposed variable usage rules HPJava

Bryan Carpenter

September 23, 1997

The proposed HPJava incorporates an idea of process groups, quite similar to the groups in, say, MPI. Groups have two essential roles. Firstly, an array or variable can be distributed or replicated over a particular group. Secondly a particular group can cooperate in a particular thread of control—its processes can be operating in loosely synchoronous "lock-step". At any point of execution in an SPMD program a particular group is singled out as the *active process group*. This is the group sharing a common thread of control with the local process. At a particular time the active process group may contain only the single local process, in which case all processes are currently acting independently. This often happens during the update of local array elements. At other times the active process group may include all physical processes. Typically this will be the case before a globally collective operation can be executed.

In general a simple variable or array element is *replicated* over some group. Again, as particular, cases this group may be the single local process or the whole set of physical processes. As a matter of definition, a variable is said to be replicated over the group active at its point of declaration. In

```
public static void main() {
  float a ;
  ...
}
```

the variable `a` is replicated over all physical processes, because when a program starts the active process group includes all processes. In

```
public static void main() {
  over(id) {
    float b ;
    ...
  }
}
```

the variable `b` is "replicated" over the single local process group. The construct `over(id)` is an idiom that splits the active process group into individual processes. We introduce it without further motivation here. As a final example the *elements* of the array

```
public static void main() {
  float c [[]] = new float [[id]] ;
  ...
}
```

are also "replicated" over local process only. The syntax illustrated declares `c` as a distributed array with a single element on each process of the active process group. (The simple rule that a variable is replicated over the process active at the point of declaration is modified for array elements.)

The proposed usage rules for HPJava are rules on how variables can be read and written in the context of a particular active process group. They are specifically designed to ensure that a variable is only read if a copy is resident on the local process, and *a variable can only be updated in a way which ensures its* value *stays the same on each process over which the variable is replicated.*

Whether the second (emphasized) aspect of the usage rules should be applied rigorously is an open question. Certainly there are examples where it is convenient to resort to the lower-level SPMD practise of declaring a variable "globally" and updating it with distinct values in distinct processes. The advantage of the strict rules is that they make it easier to detect and avoid the kind of programming errors where inconsistent values are passed to collective operations, leading to deadlocks or obscure errors.