Managing Object Placement

- System knows how to “find” objects efficiently:

  (*collection*, *index*) → *processor*

- Applications can specify a custom mapping or use simple runtime-provided options (e.g. blocked, round-robin)

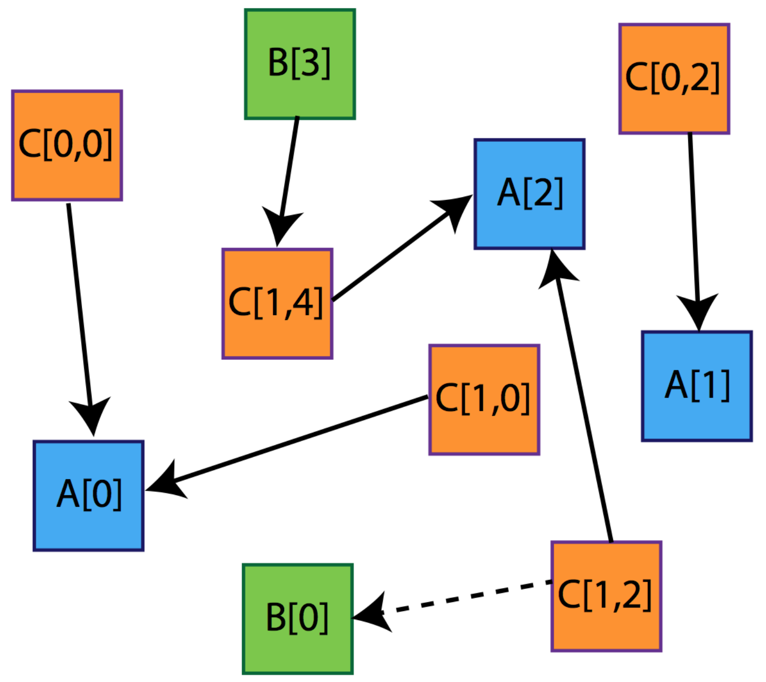
- Distribution can be static or dynamic!

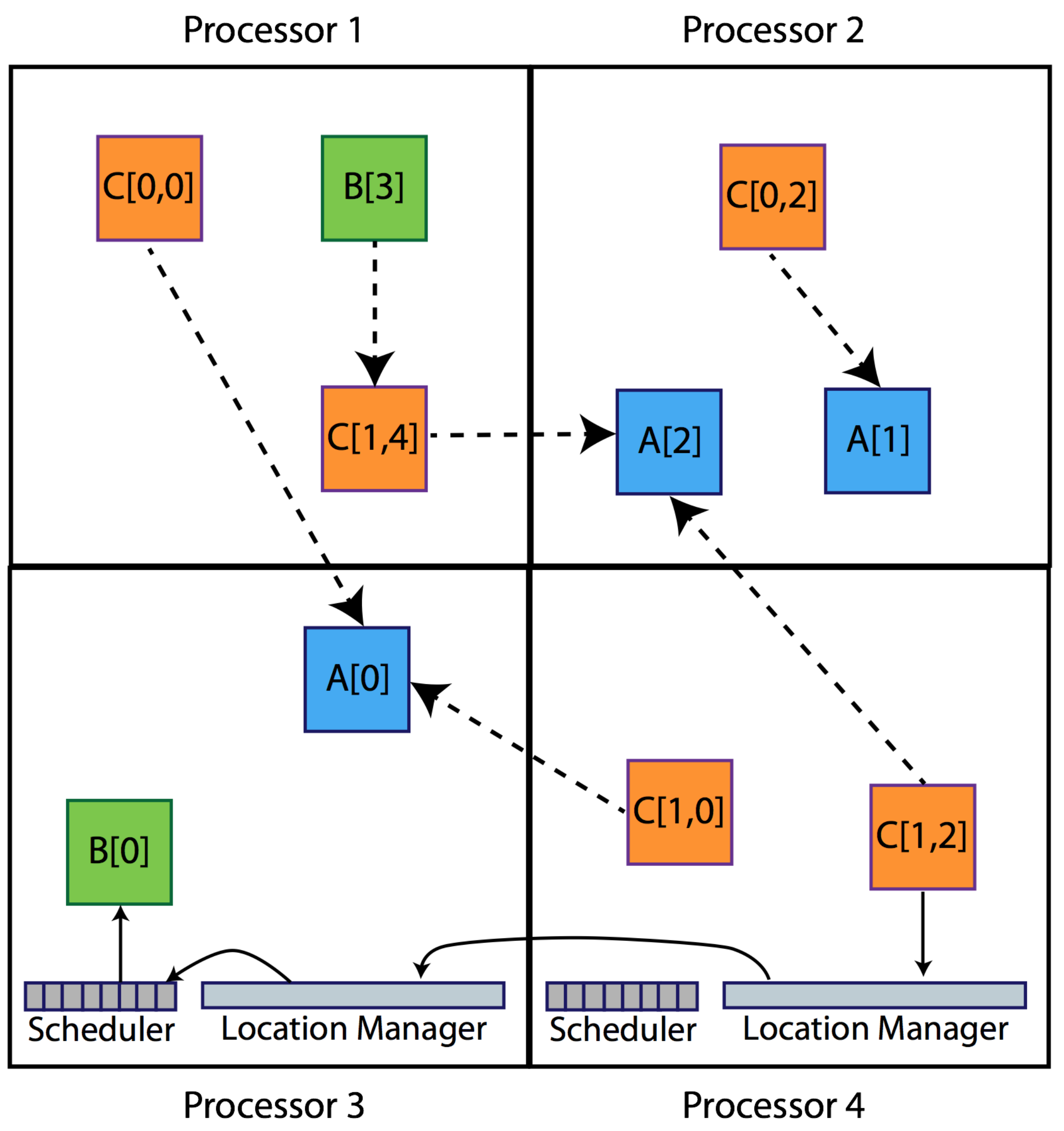
- Key abstraction: application logic doesn’t change, even though performance might

Managing Object Placement

- Application logic development decoupled from any notion of processors or object mapping
- Separation in time: make it work, then make it fast
- Division of labor: domain specialist writes object code, CS specialist writes mapping
- Portability: different mappings for different systems, scales, or configurations

Collections of Objects





Broadcast

- A message to each object in a collection
- The chare array proxy object is used to perform a broadcast
- It looks like a function call to the proxy object
- From the main chare that created a chare array:

```
CProxy_Hello helloArray =
CProxy_Hello::ckNew(helloArraySize);
helloArray.foo();
```

- From a chare array element that is a member of the same array:

```
thisProxy.foo();
```

- From any chare that has a proxy p to the chare array

```
p.foo();
```

Reduction

- Combines a set of values: `sum`, `max`, `concat`, …
- Usually reduces the set of values to a single value
- Combination of values requires an operator
- The operator must be commutative and associative
- Each object calls `contribute` in a reduction

Reduction: Example

```
mainmodule reduction {
  mainchare Main {
    entry Main(CkArgMsg* msg);
    entry [reductiontarget] void done(int value);
  };
  array [1D] Elem {
    entry Elem(CProxy_Main mProxy);
  };
}
```

Entry Method Attribute

Reduction: Example

```
#include “reduction.decl.h”
const int numElements = 49;
class Main : public CBase_Main {
public:
   Main(CkArgMsg* msg) { CProxy_Elem::ckNew(thisProxy,
numElements); }
   void done(int value) { CkPrintf(“value: %d\n”, value); CkExit(); }
};

class Elem : public CBase_Elem {
public:
   Elem(CProxy_Main mProxy) {
      int val = thisIndex;
      CkCallback cb(CkReductionTarget(Main, done), mProxy);
      contribute(sizeof(int), &val, CkReduction::sum_int, cb);
   }
};
#include “reduction.def.h”
```

Output

```
value: 1176
Program finished.
```