Task-based programming models

- Describe program / computation in terms of tasks
- Tasks might be short-lived or persistent throughout program execution
- Notable examples: Charm++, StarPU, HPX, Legion
Charm++: Key principles

- Programmer decomposes data and work into objects (called *char*es)
  - Decoupled from number of processes or cores
- Runtime assigns objects to physical resources (cores and nodes)
- Each object can only access its own data
  - Request data from other objects via remote method invocation: `foo.get_data()`
- Asynchronous message-driven execution
Hello World in Charm++

```cpp
mainmodule hello {
    array [1D] Hello {
        entry Hello();
        entry void sayHi();
    };
};
```

Charm++ Tutorial: [http://charmplusplus.org/tutorial/ArrayHelloWorld.html](http://charmplusplus.org/tutorial/ArrayHelloWorld.html)
Hello World in Charm++

mainmodule hello {
    array [1D] Hello {
        entry Hello();
        entry void sayHi();
    };
};

void Hello ::sayHi() {
    CkPrintf("Hello from chare %d on processor %d.\n", thisIndex, CkMyPe());
}

Charm++ Tutorial: [http://charmplusplus.org/tutorial/ArrayHelloWorld.html](http://charmplusplus.org/tutorial/ArrayHelloWorld.html)
Hello World in Charm++

```c++
mainmodule hello {
    array [1D] Hello {
        entry Hello();
        entry void sayHi();
    };
};

void Hello::sayHi() {
    CkPrintf("Hello from chare %d on processor %d.\n", thisIndex, CkMyPe());
}

Main::Main(CkArgMsg* msg) {
    numObjects = 5; // number of objects
    CProxy_Hello helloArray = CProxy_Hello::ckNew(numObjects);
    helloArray.sayHi();
}
```

Charm++ Tutorial: [http://charmplusplus.org/tutorial/ArrayHelloWorld.html](http://charmplusplus.org/tutorial/ArrayHelloWorld.html)
Compiling a charm program

- Charm translator for .ci file
  - Generates charm_hello.decl.h and charm_hello.def.h

  `charmcc hello.ci`

- C++ code:

  `charmcc -c hello.C`
  `charmcc -o hello hello.o`
Chare arrays

• User can create indexed collection of data-driven objects

```
CProxy_Hello helloArray = CProxy_Hello::ckNew(numElements);
```

• Different kinds: 1D, 2D, 3D, …

• Mapping of array elements (objects) to hardware resources handled by the runtime system (RTS)
Object-based virtualization

- User programs in terms of chares or objects
Over-decomposition

- Create lots of “small” objects per physical core
  - Objects grouped into arrays: 1D, 2D, …
- System assigns objects to processors and can migrate objects between physical resources
- Facilitates automatic load balancing
Message-driven execution

- An object is scheduled by the runtime scheduler only when a message for it is received
- Facilitates adaptive overlap of computation and communication
Cost of creating more objects?

- Context switch overhead
- Cache performance
- Memory overhead
- Fine-grained messages