Programming languages that incorporate parallelism

Presented by David Bunde

Knox College

Why a parallel language?

```
const D : domain(1) = [1..numRect];

proc rectArea(i : int) {
  //compute area of rectangle i
  const x = baseX + i*width;
  return width * sqrt(1.0 - x*x);
}

const halfPI = + reduce rectArea(D);
writeln(2.0*halfPI);
```

Other high-level features

- `forall` – parallel version of `for` loop
- `begin` – start asynchronous task
- `sync` variables – keep empty/full state

HJ: Extending an existing language

- `async`: create new task
- `finish`: wait for created tasks to complete

```
finish {
    async function1();
    function2();
}
```

```
public static void fib(int n) {
   if(n <= 1)
     isolated { accum += n; }
   else {
     async fib(n-1);
     fib(n-2);
   }
}
...
   finish { fib(5); }
```

Drawbacks to parallel languages

- Less low-level control
- Many are still works in progress
  - poor error messages, limited libraries and tools, ...
  - limited resources (books, tutorials, examples, ...)
- Time spent introducing new syntax

Parallel language suggestions

- As a piece of another course:
  - definitely an extension of a language students already know
- In a dedicated course:
  - make students appreciate the language by showing threads or MPI
  - focus; you probably can't cover all of even a single language