OpenMP 3.0 Feature: Error Detection Capability

Kang Su Gatlin
Visual C++
Program Manager
Why?

- OpenMP as it stands today is great for HPC
- OpenMP as it stands today is less appropriate for server side or enterprise applications
- There is simply no mechanism for error recovery – or even detection
Ideas

- We say “ideas” and not “proposals”
  - Not even half-baked
- Exception based
- Call-back function based
- Error-code based
The Problem

#pragma omp parallel
// Code here

#pragma omp barrier
// Code here

#pragma omp critical
// Code here
Idea 1: An Exception Based Approach

- Define an OpenMP Exception Class:
  ```cpp
class OMPException { ...);
```
- Use try/catch around select constructs
  ```cpp
  int foo() {
    try {
      #pragma omp parallel
      // Code here
    }
    catch (OMPException *e) {
      // Code here
    }
  }
  ```
Idea 1: Exception Based Approach

Pros
- Seems easy to implement
- Extensible
  - The exception can have info about what happened

Cons
- Only C++, not supported in C
- Can have large perf degradation
Idea 2: Error Code Based Approach

- Add a new clause to directives
- This one sets an error code in a passed address of type OMPError when error occurs

```
OMPError *ompErr = new OMPError;
#pragma omp parallel for error(ompErr)
```
Idea 2: Error Code Based Approach

**Pros**
- Also seems easy to implement
- Supports all languages
- Very general

**Cons**
- Maybe violates the “even works as expected compiled serially”
- Code to handle error is added directly to computational portion of code
Idea 3: Callback-Based Approach

- Add a new clause to directives:

```c
#pragma omp parallel error_callback(error, flag)

void error(int *flag) {
    // User code here
}
```
Idea 3: Callback-Based Approach

**Pros**
- Little performance impact if no error
- Code is kept away from site of the computation

**Cons**
- Less extensible
- Not really clear if it really does anything useful, but I like callbacks