How to give a talk

Every slide needs to have a purpose.

Goals: In general you need to effectively communicate
- the big picture problem,
- why people should care about it,
- the piece of the problem you are attacking,
- your approach,
- and an evaluation of your approach.

Project status reports will be much more succinct than a normal talk
- 10 minutes max for your presentation
- 5 minutes for class feedback
- 3 slides
- your problem
- your approach
- current status and issues

Example of Iteration Reordering

Original Loop

```plaintext
for (i=0; i<m; i++) {
    X[r[i]] = ...
}
```

Iteration reordering $T_{I_{0} \rightarrow I_{1}} = \{ [i] \rightarrow [\delta(i)] \}$

Inspector traverses the data mapping at runtime (mapping of iterations to data)

$M_{I_{0} \rightarrow X_{0}} = \{ [i] \rightarrow [r(i)] \}$

Template code for inspector (using omega library to generate loops)

```plaintext
void inspector( PARAMS ) {
    ITER_TO_DATA_LOOP_START
    ITER_TO_DATA("data_index[iter_count]")
    iter_count++;
    ...
    ITER_TO_DATA_LOOP_END
}
```

Project Overview

Goal: Efficient and scalable irregular applications
- an irregular application is one that involves indirect memory references such as X[A[i]]
- efficiency in terms of uni-processor performance, good data locality
- scalable in terms of parallel implementation

Problems
- indirect memory references do not allow static transformations
- generation of inspectors/executors to perform run-time reordering
transformations is currently ad hoc or laborious

Approach: Automatically generate inspectors and executors for composed
run-time reordering transformations.

Status

Success
- PLDI 2003 paper on framework for composing run-time reordering
transformations
- working on prototype of system that can automatically generate the
inspector using template reordering algorithms and omega code
generation for loop bounds

Issues
- still developing a way to automatically generate the executors
- index arrays with complex relationships will make iteration reordering
difficult

```plaintext
for (i=0; i<N; i++) {
    for (p=ia[i]; p<ia[i+1]; p++) {
        X[p] = ...;
        X[i] = ...;
    }
}
```