Almost Free Concurrency! (using GOF patterns)

Sean L. Mooney, Hridesh Rajan, Steven M. Kautz, Wayne Rowcliffe

Problem: Scalability of General-purpose OO Applications

- Not Embarrassingly Parallel (unlike scientific applications)
  - Irregular parallelism
  - Parallelizing compilers don’t work very well
- Parallel patterns [Schmidt, Matteo] require concurrency expertise
  - Implicitly parallel languages require changing tools
- Explicitly Adding Concurrency HARD and ERROR PRONE!
  - Where can it go?
  - Was it done correctly?
  - Causes tangling of concurrency concern with other concerns.
- ... But these Apps also Require Concurrency to Scale now

Solution: Exploit Well-defined Gang-of-Four Patterns

- WHAT: Concurrent GOF Pattern Framework
  - Contains implicitly concurrent versions of GOF patterns
- HOW: Exploit structure of the GOF design patterns
  - Design patterns define interactions between participants
  - Some of these interactions can be potentially concurrent
  - Expose them for implicit concurrency between participants
- WHY: Modularity + Concurrency at the same time
  - Separate and encapsulate the concurrency concern
  - Reduce use of explicitly concurrent features, avoid errors
  - Let our framework worry about your concurrency concerns

Pattern Examples

- **Decorator Pattern**
  
  ```
  t = new OriginalType();
  da = new DecoratorA(t);
  db = new DecoratorB(db);
  db.mainBehavior();
  ```

- **Builder Pattern**
  
  ```
  Grader g = AsyncUtilities.createAsyncBuilder(new Grader());
  g.grade(student1);
  g.grade(student2);
  ... g.grade(student_N);
  g.getGradeReport();
  ```