Designing Scalable and Creative Algorithms

Elliott Sprehn (Google, Inc.)
esprehn@gmail.com
http://www.elliottsprehn.com/blog/
http://www.elliottsprehn.com/cfbugs/

Algorithms

- Procedures for solving problems
- Common (Classical) Problems
  - Sorting
  - Merging
  - Queries
- Every Day Problems
  - ex. Combine several XML files

Analysis Toolbox

- Categorizing Algorithms
  - Theoretical
    - Big-O Notation
    - Disk Accesses
  - Practical
    - “Work” Done
    - Hot Spots

Big-O

- Mathematical Notation
- Standard for Algorithm Analysis
  - O(1) Constant Time
  - O(n) Linear Time
  - O(n^2) Polynomial Time (Quadtratic)

Code Sample:

```
function sum100(array) {
    var value = 0;
    for(var i=1; i lte 100; ++i)
        value += array[i];
    return value;
}
```

```
function employeeHierarchyList(employee) {
    for( var list = [employee];
        employee.hasParent();
        employee = employee.getParent() )
         arrayAppend(list,employee.getParent());
    return list;
}
```

Disk Accesses

- Disk is **extremely** slow
- Measure algorithm by disk accesses element
  - Difficult, your OS has lots of optimization

Code Sample:

```
function writeNumbers(fileName,n) {
    var handle = fileOpen(fileName,”write”);

    // How many disk writes is this?
    try {
        for(var i=1; i lte n; ++i)
            fileWrite(handle,i);
    } finally {
        fileClose(handle);
    }
}
```

Lets be Practical

- **Theoretical** Measurements
  - Difficult to use in real world situations
  - O(2n) is the same as O(500n)
  - $O(n^2)$ is faster than O(500n) for small n

“Work” Done

- Extremely Rough Estimate
- Count steps per unit
  - Minimize the number of steps
- Problem: Not all steps are equal

Hot Spots

- Steps that take longer
- Find hot spots
  - Remove or Reduce

Code Sample:

```
var values = {};
for( var i =1; i <= n; ++i ) {
  var value = randRange(0,100);
  if( structKeyExists(values,value) ) {
      values[createUUID()] = true;
  } else {
      values[value]  = true;
  }
}
return structKeyArray(values);
```

Design Toolbox

- Structs
- Arrays
- Queries
- Function Pointers
- Components

Sorting

- Classical Problem
  - Often handled by SQL in every day applications
  - Trust the API, it's faster
- Conventional Algorithm Choices
  - Not very relevant in most applications
  - QuickSort
  - MergeSort
  - ...

Choose the Right Format to Sort

- Arrays
  - ArraySort()
  - QuickSort() (CFLib)
- Structs
  - StructSort()
- Queries
  - QoQ order by

Arrays

- ArraySort()
  - Easy to sort simple values
- QuickSort()
  - Implement yourself (Wikipedia)
  - Get from CFLib
  - Uses a callback function

Lets create a sorting algorithm...

Merging

- Another Classical Problem
  - Handled in SQL with **union** and **join**
- What do we merge?
  - Queries
  - Arrays
  - Lists
  - Collections of objects
- How to define concept of merged?
  - Sorted, Unique?

Choose the right Format to Merge

- Arrays
  - Several Algorithms
- Structs
  - StructAppend()
  - Concept of “merged” difficult.
- Queries
  - QoQ union, join

Lets create a merge algorithm...

Creative Algorithm Examples

- CFUnited Advisory Board Application
  - Each Topic Worksheet stored in XML
  - Need to merge and sort the data

- Transfer ORM
  - ObjectManager.getObject()
    - Creates the object graph of ORM type definitions
    - Need to efficiently create the graph
    - Must be thread safe
  - SelectStatement.executeEvaluation()
    - Uses cfquery to execute a compiled query
    - Need to use cfqueryparam in arbitrary places in generated code
  - XMLFileReader.search()
    - Find configuration information in transfer.xml file
    - Handle imports and includes in XML

Using Caching (Space vs Speed)

- Trades memory for “speed”
- Makes algorithm analysis difficult
  - Cache hits/misses depends heavily on data
  - How much “work” is saved by using the cache?
- Cache Hot Spots
  - Often provides the most benefits
- Cache Frequented Paths

Caching Examples

- Accelerate Framework
  - Routing
    - Compiles “routes” to regular expressions
    - Generates URLs for routes (slow matching process)

- Shared Application Architecture
  - getObject(name)
    - Calls ColdSpring getBean(name)
    - Not a Hot Spot
    - Inefficient locking inside ColdSpring
    - Called very often!

- CFUnited 2010 Website
  - Schedule and Speakers queries are expensive
  - Cache our RecordSet object with state

Questions