SExtractor as a Web Service

Presented by:

Craig D. Miller

JPL

July 13, 2004

What is SExtractor?

- SExtractor is a neural net program that builds a catalog of objects from an astronomical image
- Originally written by Emmanuel Bertin
- Now in the public domain at:
  - http://sourceforge.net/projects/sextractor/

SExtractor as a Web Service

Why change working program into a web service?

- Allow user to not worry about implementation details
- Let Workflow applications control multiple services
- I'll not further answer this question (Roy Willams' & Joe Jacob's [and other's] talks hopefully address this)

SExtractor as a Web Service

program analysis

- First one needs to understand the program
  - What are the program inputs?
    - A FITS image
    - Define search criteria (or use defaults)
  - What are the program outputs?
    - Catalog of found objects
  - What language was it written in?
    - Sextractor is written in C

SExtractor as a Web Service

Design Decisions

- Create a web service using Tomcat
  - An open source implementation of Java Servlet and JavaServer
- Use Axis under Tomcat
  - Basically Apache SOAP 3.0 implementation (Simple Object Access Protocol)
- Use JAVA JNI (Java Native Interface)
  - Allows us to use C source without too many changes
  - Can write C wrapper to convert Java input into a command line

SExtractor as a Web Service

Creating a generic Tomcat/Axis web service

- Write simple interface description and compile

```
/**
 * Sextractor.java
 */
package grist.Sextractor;

public class Sextractor {
    public static byte[] sextractor (byte[] in0) throws Exception
    {
        return in0;
    }
}
```

- run java2WSDL to create wsdl interface description
- run WSDL2java to create server code
- Populate server code with actual service code
- Compile and deploy service under Tomcat/Axis
- Write client and compile (generate from WSDL)
- Run client to get results

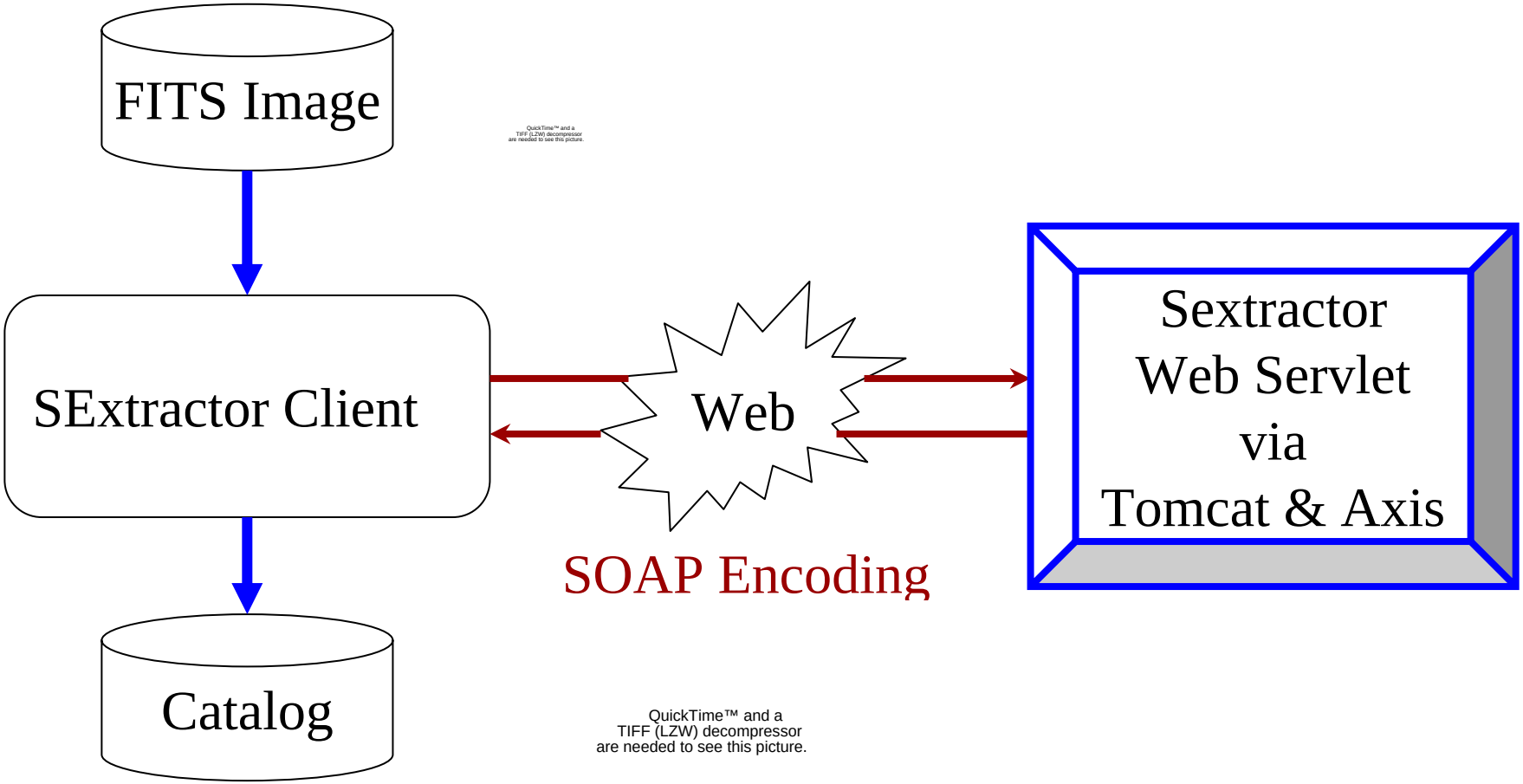
SExtractor as a Web Service

Converting Sextractor program to a JNI library

- Global C variables must be initialized
- JNI wrapper code needs to convert Java input for library
- Need to change name of main() entry point
- Need to change any exit (& error) code to return back to JAVA (and preferably any errors throw an exception)
- Place any default input files in class directory for ease of maintenance and locating

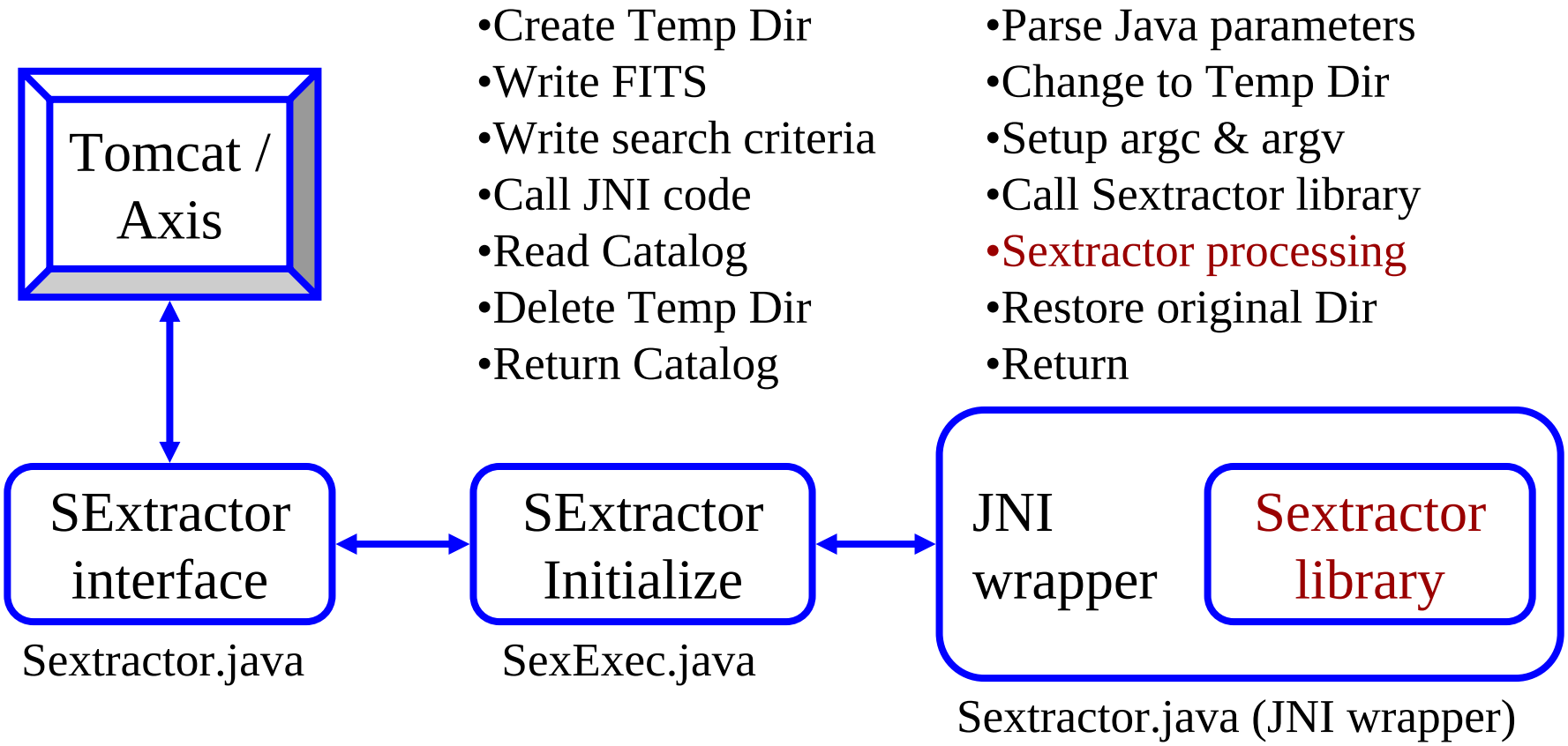
SExtractor as a Web Service

Client Server Data Flow



SExtractor as a Web Service

Server Data Flow



SExtractor as a Web Service

Overview (Java code & classes)

- grist.Sextractor (simple interface seen by outside)
  - Client (SexClient)
  - Servelet (Sextractor interface & Servlet code)
- grist.Sextractor.SexExec (Interface and setup for JNI code)
  - SExtractor setup code (SexExec)
  - JNI Wrapper
  - SExtractor default input files
  - SExtractor library (C code)
- grist.Sextractor.SexExec.sextractor-2.3.2 (C code for library)
  - javamain.c - C code wrapper to setup argc & argv so library thinks it's still a program run from command line
  - Rest of original code with minor modifications

SExtractor as a Web Service

SExtractor Client

```
/**
 * SexClient.java
 */

import java.io.*;
import grist.Sextractor.*;

public class SexClient {
  public static void main(String[] args) throws Exception {
      String endpoint =
            "http://mach.jpl.nasa.gov:8080/axis/services/Sextractor";

      //Usage
      if (args.length < 2)
      {
         System.out.println("Usage: java SexClient in.fits out.cat");
         return;
      }

      // Read input file so we can pass to Sextractor
      BufferedInputStream inFile = new BufferedInputStream(
                  new FileInputStream( args[0] ) );
      byte [] inData = new byte [ inFile.available() ];
      int i = inFile.available();
      System.out.println("inFile.available() = " + i);
      int len = inFile.read( inData );
      System.out.println("inFile size = " + len);
      inFile.close();

      // See if we can create output file
      BufferedOutputStream outFile = new BufferedOutputStream(
                  new FileOutputStream( args[1] ) );

      // Now call Sextractor web service
      SextractorService service = new SextractorServiceLocator();
      Sextractor stub = service.getSextractor (new
        java.net.URL(endpoint));
      byte [] result = stub.sextractor(inData);
      System.out.println("Got result.length : " + result.length);

      // Output results to file
      System.out.println("Saving results to : " + args[1]);
      outFile.write(result, 0 , result.length);
      outFile.close();
   }
}
```

SExtractor as a Web Service

SexExec (main server code)

```
/**
 * SexExec.java
 */
package grist.Sextractor.SexExec;
import java.io.*;
import grist.Sextractor.*;
import grist.Sextractor.SexExec.*;
import grist.Sextractor.SexExec.Sextractor;

public class SexExec {

   // HELPER ROUTINES REMOVED FOR BREVITY

   public byte[] sextractor (byte[] fitsData) throws Exception
   {

      // Create a temporary working directory for Sextractor
      File tempDir = createTempDir("sex",".tmp");
      String tempDirName = "" + tempDir;
      //System.out.println("Use Dir:" + tempDirName);

      //Save fitsdata into fits file in temp dir
      String fName = "check.fits";
      String fitsName = tempDir + File.separator + fName;
      BufferedOutputStream outFile = new BufferedOutputStream(
                  new FileOutputStream( fitsName ) );
      outFile.write(fitsData, 0 , fitsData.length);
      outFile.close();

      // Copy default.param to temp dir
      String defParam = "default.param";
      copyFile(defParam,
            tempDir + File.separator + defParam);

      // Copy default.conv to temp dir
      String defConv = "default.conv";
      copyFile(defConv,tempDir + File.separator + defConv);

      // Copy default.sex to temp dir
      String defSex = "default.sex";
      copyFile(defSex,tempDir + File.separator + defSex);

      // Execute Sextractor
      String params[] = { tempDirName, fName};
      Sextractor sex = new Sextractor();
      int ret = sex.evaluate(params);

      //Read test.cat (from temp dir)
      String catFile = tempDir + File.separator + "test.cat";
      BufferedInputStream inFile = new BufferedInputStream(
                  new FileInputStream( catFile ) );
      byte [] catData = new byte [ inFile.available() ];
      int i = inFile.available();
      int len = inFile.read( catData );
      inFile.close();
```

SExtractor as a Web Service

SexExec (main server & JNI code)

```
 /**
 * SexExec.java (CONTINUED)
*/
      // And finally delete any files (& directories) that were created
      if(!deleteDir(tempDir)) throw new IOException();

       return catData;
   }
}
```

```
/**
 * Sextractor.java (JNI Interface to native C library)
*/
package grist.Sextractor.SexExec;

import java.net.URL;

public class Sextractor {
  static {
    URL url = Sextractor.class.getResource("libsex.so");
    String Lib = "" + url;          //Lib is name of lib with fill path
    System.load(Lib.substring(5));  //Note: need to strip URL "file:" off

    //Disable code that required LD_LIBRARY_PATH
    //System.loadLibrary("sex");
  }
  public native int evaluate(String[] args);
}
```

SExtractor as a Web Service

javamain.c (JNI Library wrapper & Entry point)

```
 /*
javamain.c
*/

#ifdef __cplusplus
extern "C" {
#endif
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "Sextractor.h"
#include "fits/fitscat.h"

int     javaMain(int argc, char *argv[]);

/*
 * Class:     Sextractor
 * Method:    evaluate
 * Signature: ([Ljava/lang/String;)V
 */
JNIEXPORT jint JNICALL
        Java_grist_Sextractor_SexExec_Sextractor_evaluate
  (JNIEnv *env, jobject jobj, jobjectArray jarray)
{
   /* Obtain the size the the array with a call to the JNI function
        GetArrayLength() */
   jsize Cargc = (*env)->GetArrayLength(env, jarray);
```

```
/* Declare a char array for argv */
#define MAXARGS 128
   char *Cargv[MAXARGS];
   int i;
   // Declare a char array for temp dir
   char *tdir = NULL;
   char *cwd = NULL;

   /* Check number of arguments */
   if (Cargc >= MAXARGS)
   {
      fprintf(stderr, "ERROR too many arguments");
      //DEBUG need to throw java exception here
      return 1;
   }

   /* Get current working directory */
#if ! defined PATH_MAX
# define PATH_MAX 4096
#endif
   if ((cwd = getcwd(NULL, PATH_MAX)) == NULL) {
      fprintf(stderr, "ERROR cannot get current working directory");
      //DEBUG need to throw java exception here
      return 2;
   }
#define DEBUG_JAVA 0
#if DEBUG_JAVA
   printf("DEBUG cwd = %s\n", cwd);
#endif
```

SExtractor as a Web Service

javamain.c (JNI Library wrapper & Entry point)

```
  //Cargv[0] = "JavaMain"; //So program won't bomb if it accesses Argv[0]
  // when i=1 we have temp dir name
  for (i = 1; i < Cargc + 1; i++)
  {
    /* Obtain the current object from the object array */
    jobject myObject = (*env)->GetObjectArrayElement(env, jarray, i-1);

    /* Convert the object just obtained into a String */
    const char *str = (*env)->GetStringUTFChars(env, myObject, 0);

    if (i == 1)
    {
      tdir = malloc(strlen(str));
      strcpy(tdir, str);
    }
    else
    {
       /* Build the argv array */
       Cargv[i - 1] = malloc(strlen(str));
       strcpy(Cargv[i - 1], str);
#if DEBUG_JAVA
       printf("DEBUG argv %d = %s\n", i-1, Cargv[i - 1]);
#endif
    }

    /* Free up memory to prevent memory leaks */
    (*env)->ReleaseStringUTFChars(env, myObject, str);
  }
```

```
/* Change to temporary directory */
  if (chdir(tdir)) {
    fprintf(stderr, "ERROR cannot change to %s directory", tdir);
    //DEBUG need to throw java exception here
    return 3;
  }
#if DEBUG_JAVA
  printf("DEBUG tdir = %s\n", tdir);
#endif

  /* Increment argc to adjust the difference between Java and C arguments */
  //Not needed, as first argument is now temp directory name
  //Cargc++;
#if DEBUG_JAVA
  printf("DEBUG argc = %d\n", (int) Cargc);
#endif

  /* Call Sextractor main function which uses command line arguments */
  printf("C Before call\n");
  i = javaMain((int)Cargc,Cargv);
  printf("C After call\n");

  /* Change back to default directory */
  if (chdir(cwd)) {
    fprintf(stderr, "ERROR cannot change to %s directory", cwd);
    //DEBUG need to throw java exception here
    return 4;
  }

  return i;
}
#ifdef __cplusplus
}
#endif
```

SExtractor as a Web Service

What's next?

- Fix minor code issues (test/fix all exits/errors)
- Allow client to send default overrides
- Change to use VOTables
- Stateful web server
- Standardize "dictionary" methods for inputs, outputs, and execution for this & other web services
- Authentication and Security
- Large FITS image issues
- Workflow integration - Triana, or ???
- Numeric differences on different platforms