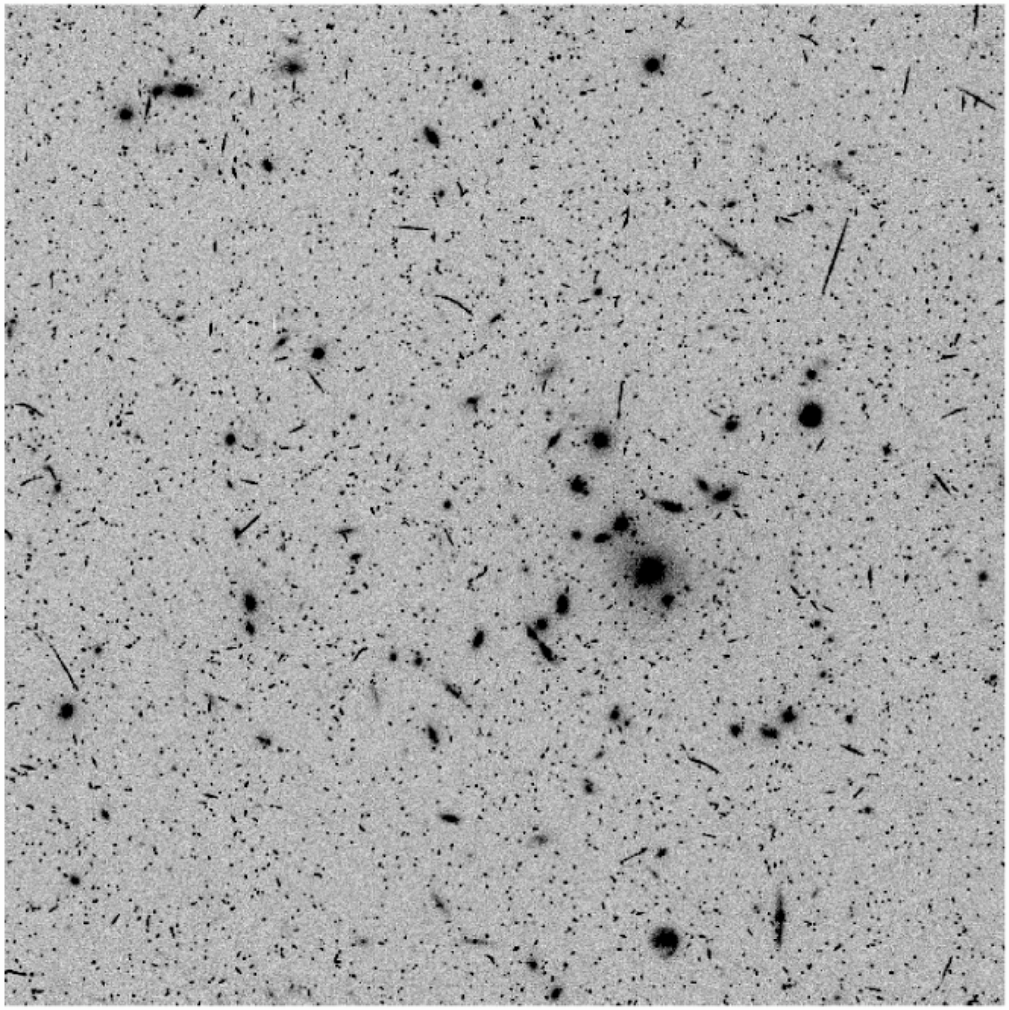
CRBLASTER: Benchmarking a Cosmic-Ray Rejection Application on the Tilera 64-core TILE64 Processor

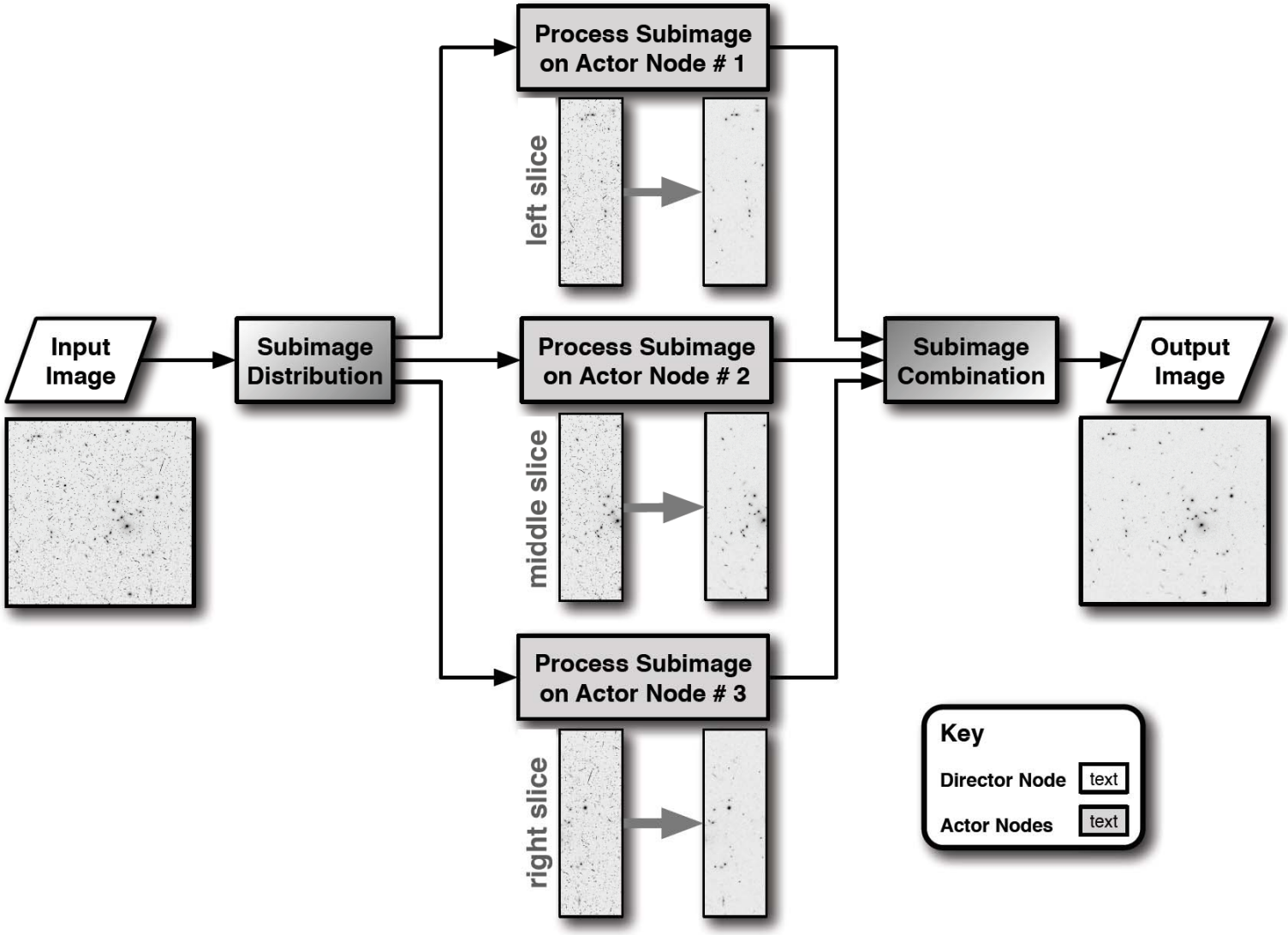
Kenneth Mighell
National Optical Astronomy Observatory

High Performance Embedded Computing Workshop 2010
MIT Lincoln Laboratory
September 15, 2010



WF3 dataset of the 2400-s HST WFPC2 observation
U3060302M.C0H of the galaxy cluster MS 1137-67

CRBLASTER Parallelization



PUBLICATIONS OF THE ASTRONOMICAL SOCIETY OF THE PACIFIC, **113**:1420–1427, 2001 November

Cosmic-Ray Rejection by Laplacian Edge Detection

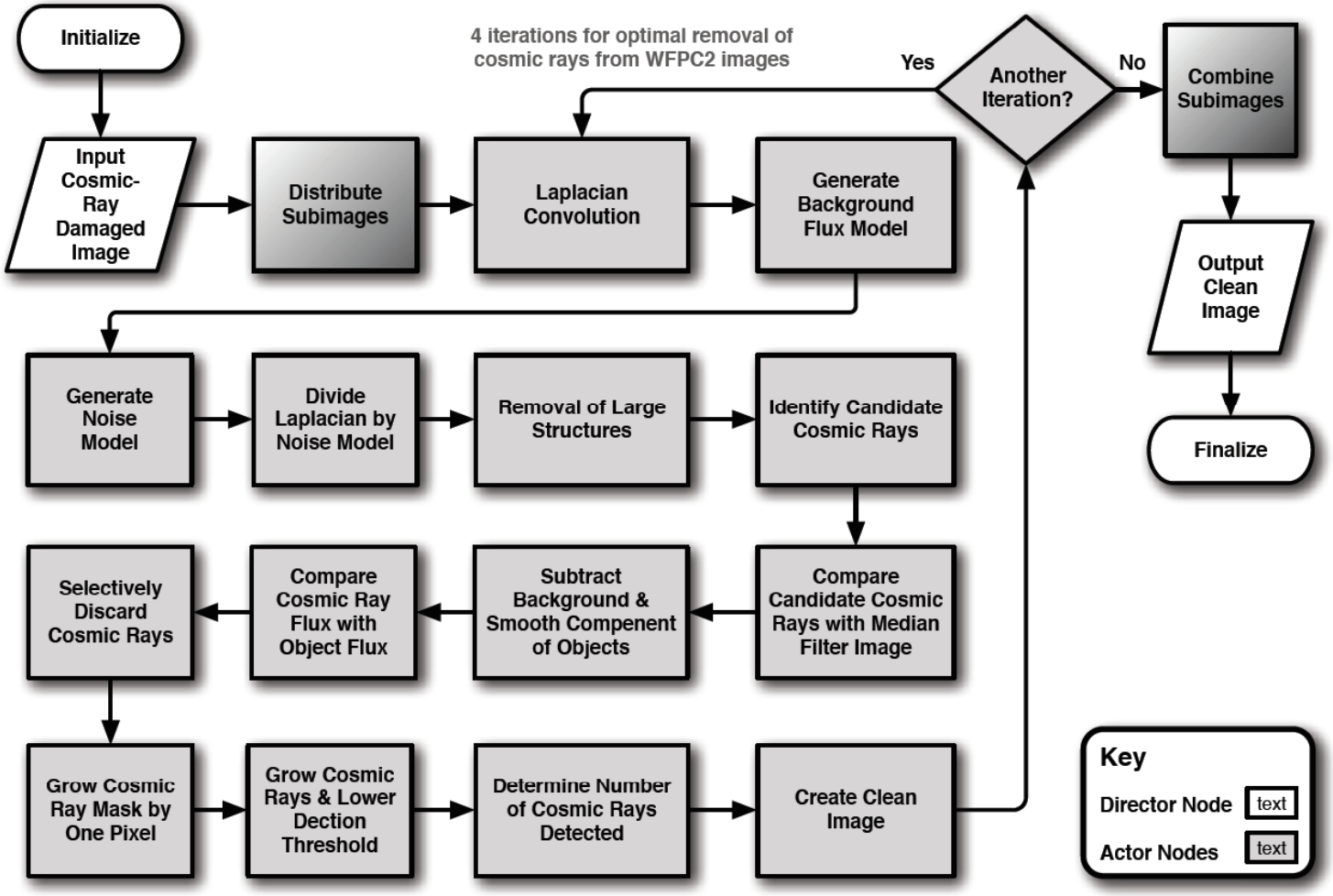
PIETER G. VAN DOKKUM[1]

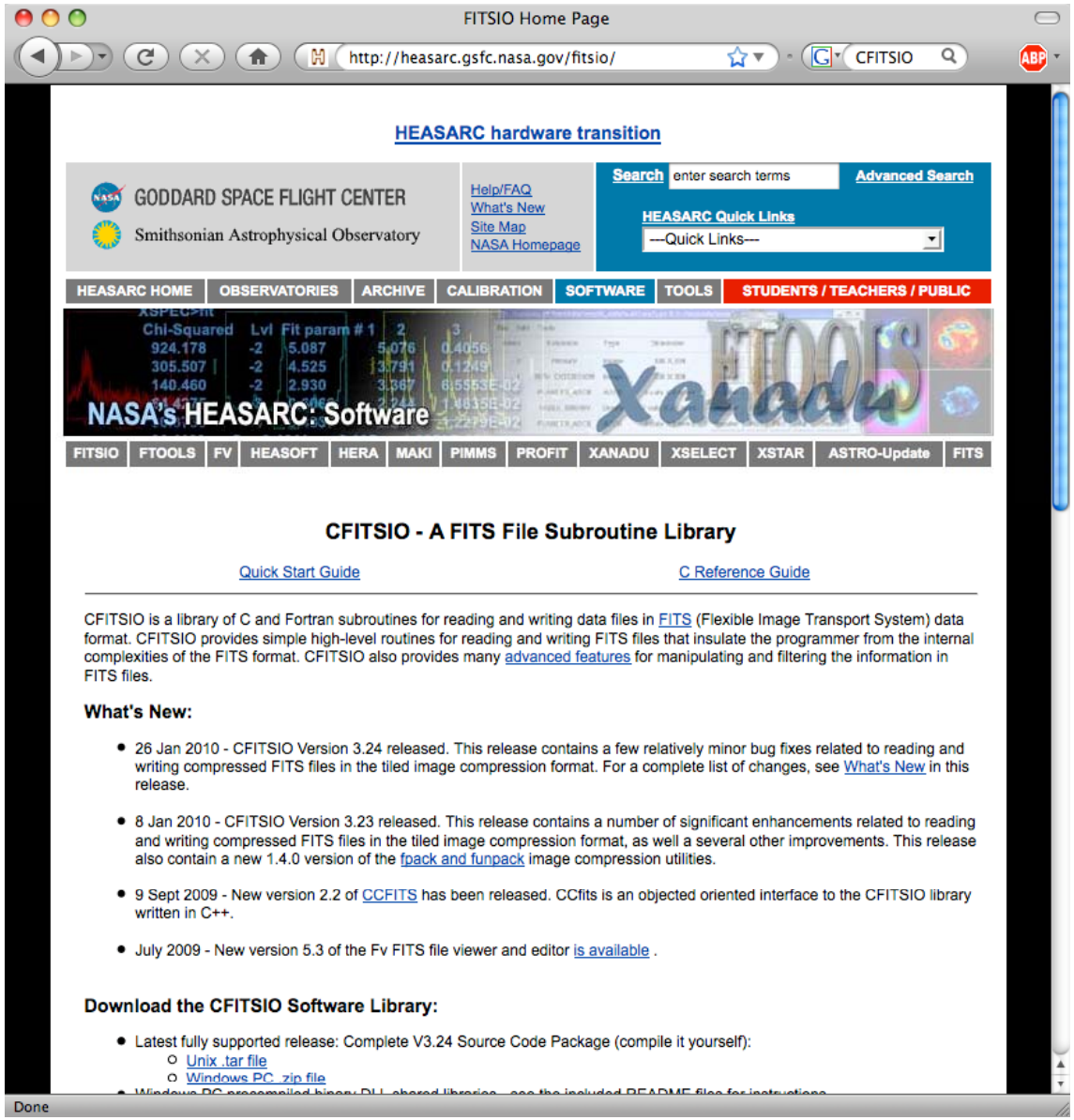
California Institute of Technology, MS 105-24, Pasadena, CA 91125; pgd@astro.caltech.edu

Received 2001 May 1; accepted 2001 July 31

ABSTRACT. Conventional algorithms for rejecting cosmic rays in single CCD exposures rely on the contrast between cosmic rays and their surroundings and may produce erroneous results if the point-spread function is smaller than the largest cosmic rays. This paper describes a robust algorithm for cosmic-ray rejection, based on a variation of Laplacian edge detection. The algorithm identifies cosmic rays of arbitrary shapes and sizes by the sharpness of their edges and reliably discriminates between poorly sampled point sources and cosmic rays. Examples of its performance are given for spectroscopic and imaging data, including *Hubble Space Telescope* Wide Field Planetary Camera 2 images.

CRBLASTER Flowchart Diagram




FITSIO Home Page
http://heasarc.gsfc.nasa.gov/fitsio/
CFITSIO
HEASARC hardware transition
GODDARD SPACE FLIGHT CENTER
Smithsonian Astrophysical Observatory
Help/FAQ
What's New
Site Map
NASA Homepage
Search
enter search terms
Advanced Search
HEASARC Quick Links
---Quick Links---
HEASARC HOME
OBSERVATORIES
ARCHIVE
CALIBRATION
SOFTWARE
TOOLS
STUDENTS / TEACHERS / PUBLIC
NASA's HEASARC: Software
FITSIO
FTOOLS
FV
HEASOFT
HERA
MAKI
PIMMS
PROFIT
XANADU
XSELECT
XSTAR
ASTRO-Update
FITS
CFITSIO - A FITS File Subroutine Library
Quick Start Guide
C Reference Guide
CFITSIO is a library of C and Fortran subroutines for reading and writing data files in FITS (Flexible Image Transport System) data format. CFITSIO provides simple high-level routines for reading and writing FITS files that insulate the programmer from the internal complexities of the FITS format. CFITSIO also provides many advanced features for manipulating and filtering the information in FITS files.
What's New:
26 Jan 2010 - CFITSIO Version 3.24 released. This release contains a few relatively minor bug fixes related to reading and writing compressed FITS files in the tiled image compression format. For a complete list of changes, see What's New in this release.
8 Jan 2010 - CFITSIO Version 3.23 released. This release contains a number of significant enhancements related to reading and writing compressed FITS files in the tiled image compression format, as well a several other improvements. This release also contain a new 1.4.0 version of the fpack and funpack image compression utilities.
9 Sept 2009 - New version 2.2 of CCFITS has been released. CCfits is an objected oriented interface to the CFITSIO library written in C++.
July 2009 - New version 5.3 of the Fv FITS file viewer and editor is available .
Download the CFITSIO Software Library:
Latest fully supported release: Complete V3.24 Source Code Package (compile it yourself):
Unix .tar file
Windows PC .zip file
Done

Build the CFITSIO library

```
tar -xvf cfitsio3090.tar
cd cfitsio
tile-monitor --pci --mount-tile /usr --here
run ./configure
run make
quit
cd ../
```

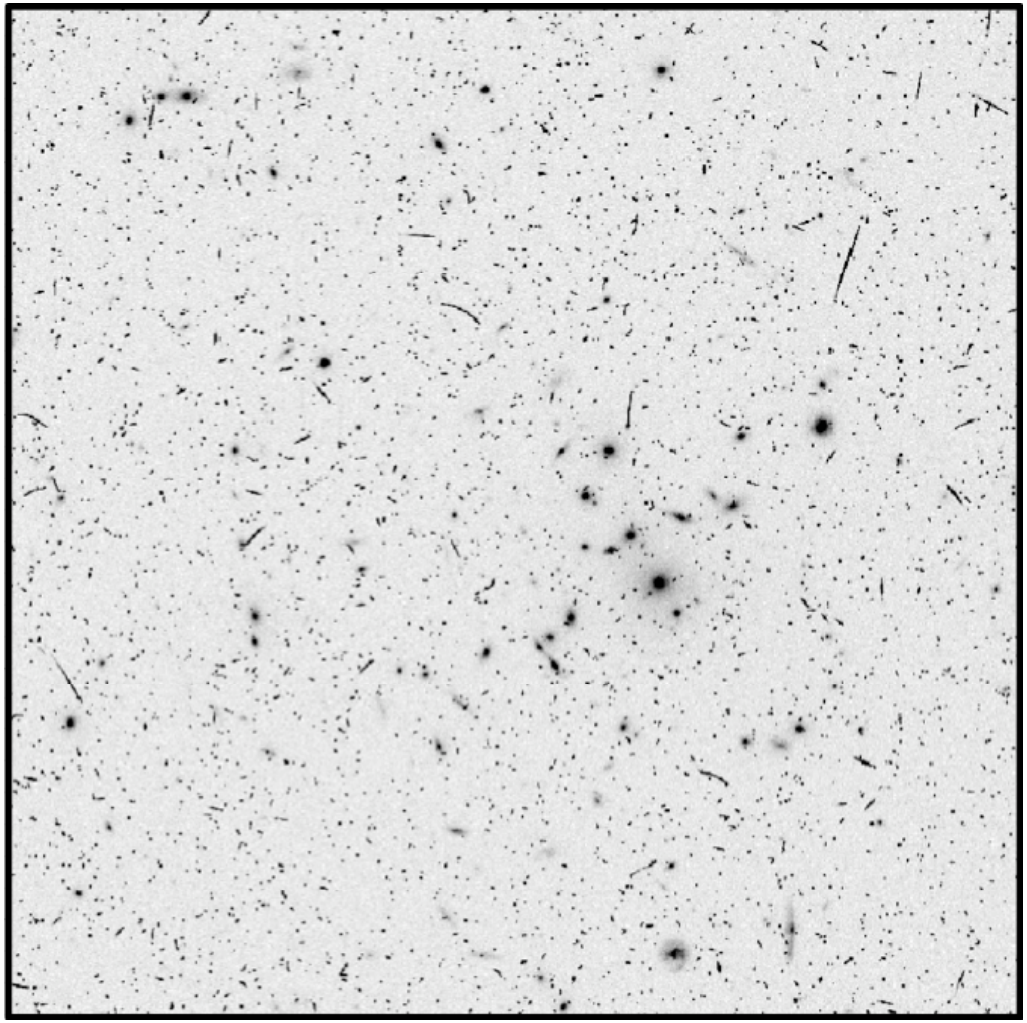
Compile ***natively*** on TILExpress-20G card

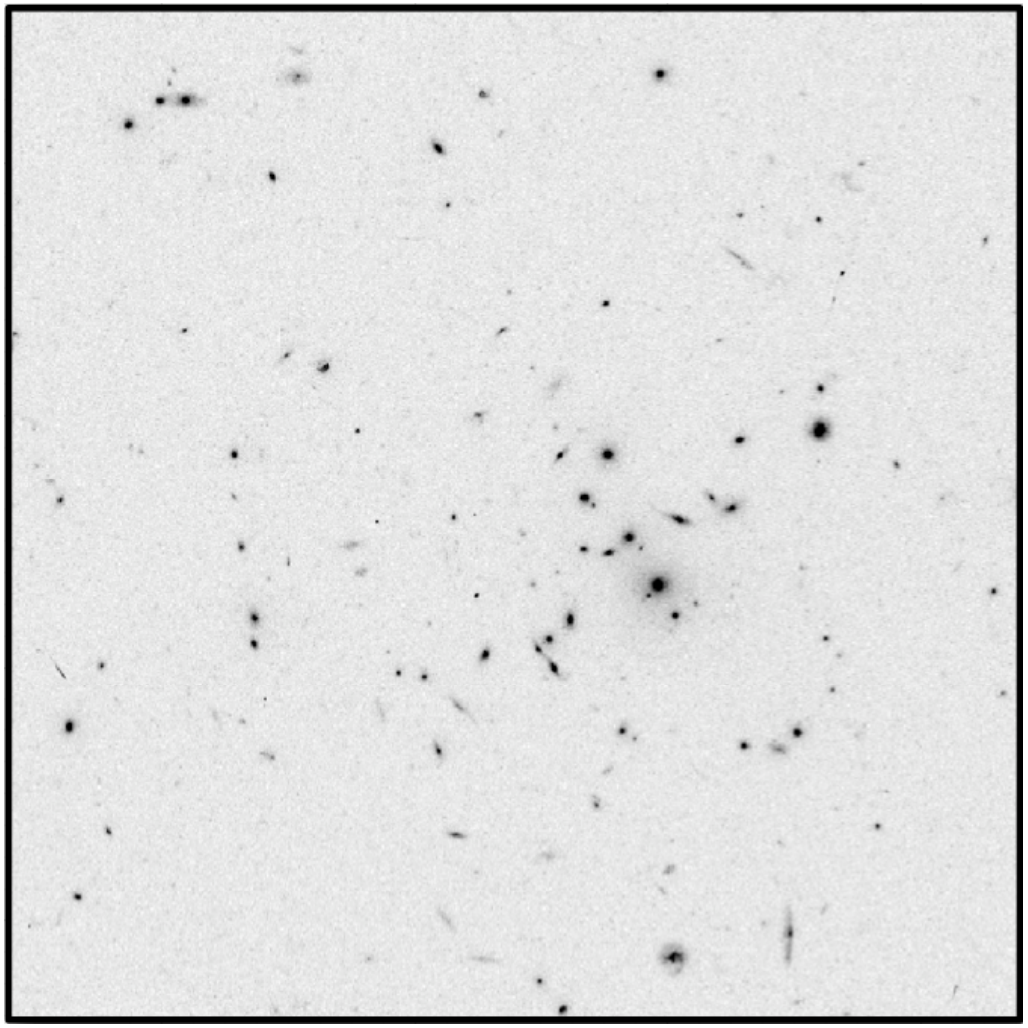
I have added the following lines to fitsio2.h:

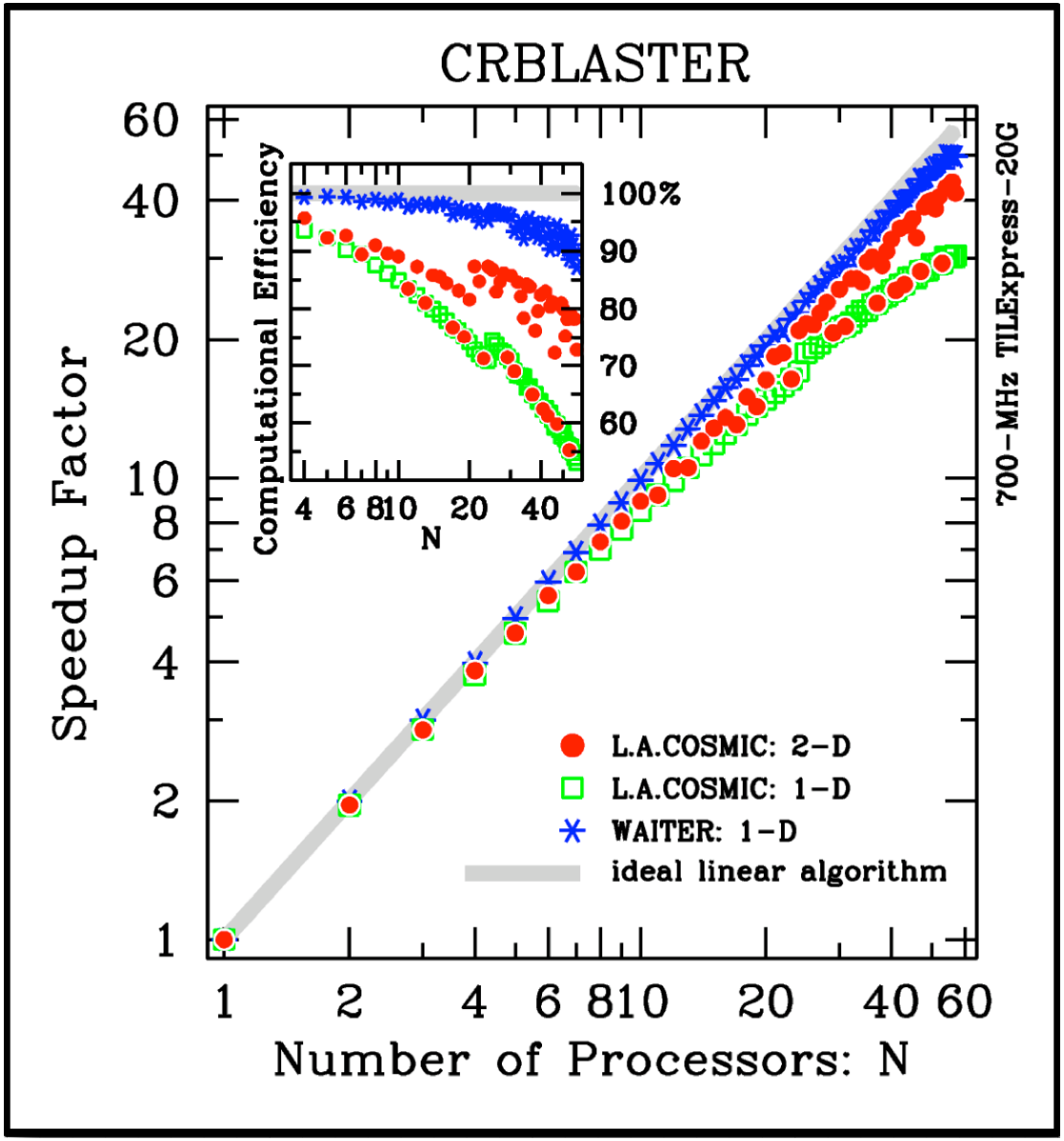
```
/* MIGHELL 2009SEP23: Tilera Tile64 processor values */
#ifdef __tile__
#undef  MACHINE
#define MACHINE  OTHERTYPE
#undef  BYTESWAPPED
#define BYTESWAPPED TRUE
#undef  LONGSIZE
#define LONGSIZE 32
#endif
```

This information tells CFITSIO that the Tile64 processor requires parts of the CFITSIO code to do byteswapping and that the size of C long variables is 32 bits. With this modest addition to fitsio2.h CFITSIO successfully builds natively on the Tile64 processor and produces correct results.

```
#      FILE: Makefile
#   PURPOSE: Build CRBLASTER on TILE64 processor (TILExpress-20G card)
#    AUTHOR: Kenneth J. Mighell [ mighell at noao dot edu ]
#  LANGUAGE: ANSI
#      DATE: 2010AUG30
# COPYRIGHT: (C) 2010 Kenneth John Mighell
#
#   WARNING: Assumes that cfitsio/libcfitsio.a exists for TILE64 processor
#

BIN= $(TILERA_ROOT)/bin
MPI= $(TILERA_ROOT)/mpi
CFITSIODIR= ./cfitsio
LIBS = -L$(CFITSIODIR) -lcfitsio -lm -lc
INCLUDE= -I./ -I$(CFITSIODIR)
OPTIMIZATION= -O2
EXTRA_CFLAGS= -Wall -pedantic $(OPTIMIZATION)
TILERA_CFLAGS= -xc -g -lm -OPT:alias=restrict -I$(MPI)
CFLAGS= -std=c99 $(EXTRA_CFLAGS) $(TILERA_CFLAGS)
CC = $(BIN)/tile-cc
CCC = $(CC) $(CFLAGS) $(INCLUDE)
LDFLAGS = -L$(MPI) -static -lmpi -ltmc -lm
LD=$(CC)
SOURCES = sort.c util.c images.c fits.c lacosc.c poisson.c main.c
INCS = sort.h util.h images.h fits.h lacosc.h poisson.h
OBJECTS = $(SOURCES:.c=.o)
TARGET = crblaster

#rules:
.c.o: $*.c $(INCS)
        $(CCC) -c $*.c

all: $(TARGET)

$(TARGET): $(OBJECTS) $(INCS)
        $(LD) -o $@ $(OBJECTS) $(LIBS) $(LDFLAGS); \
        echo compiler used: `which $(CC)`

clean:
        /bin/rm -f $(TARGET) $(OBJECTS)
```



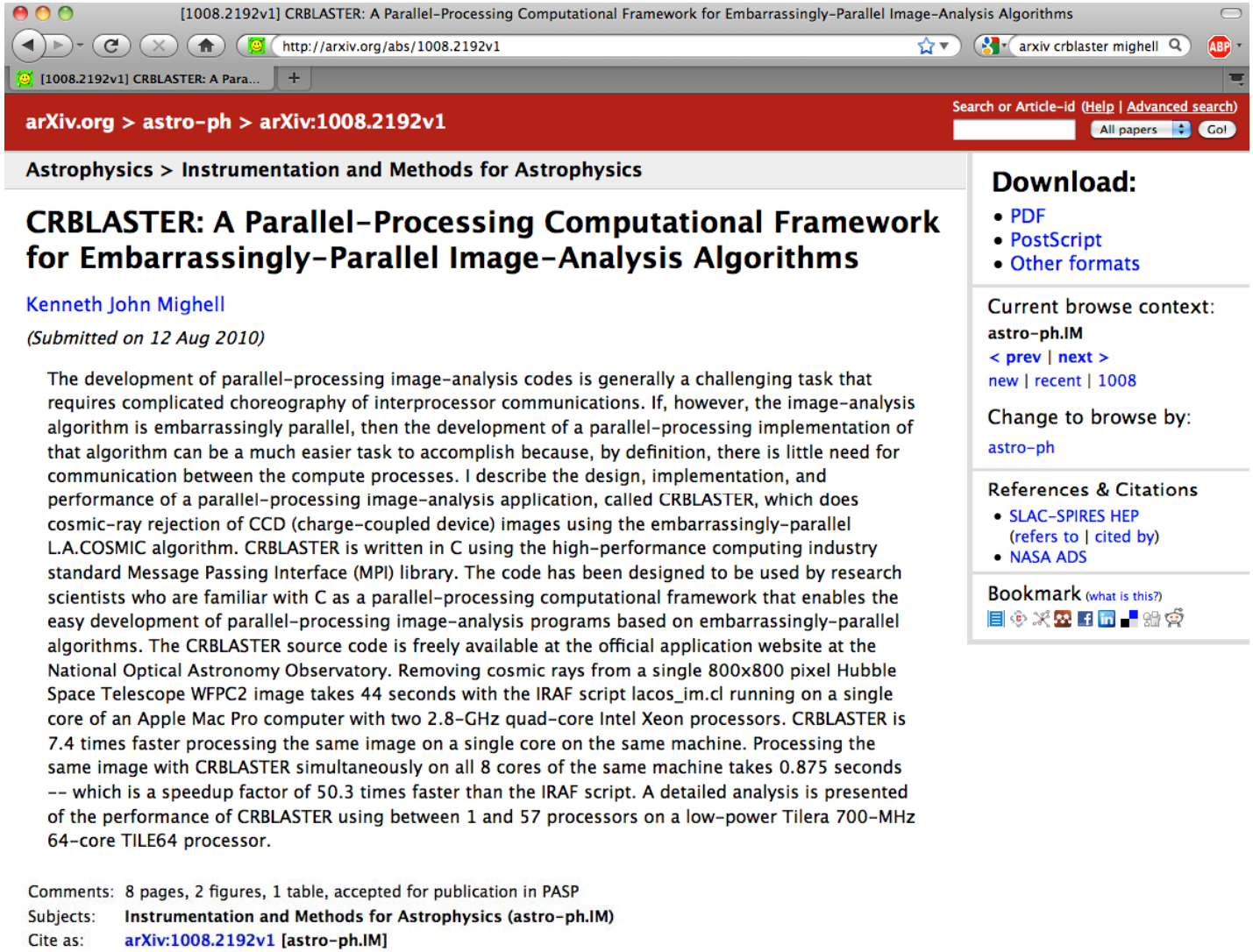




Mighell, K. J. 2010, PASP (in press: October 2010)

N	L.A.COSMIC 1-D (%)	L.A.COSMIC 2-D (%)	POISSON 1-D (%)	WAITER 1-D (%)
1	100.00	100.00	100.00	100.00
2	97.84	97.87	99.83	99.79
3	95.00	94.95	99.76	99.68
4	93.59	95.70	99.47	99.36
5	92.29	92.28	99.53	99.45
6	90.18	92.69	99.37	99.31
7	89.41	89.43	98.65	98.56
8	87.56	90.99	98.96	98.92
9	86.07	89.56	98.49	98.30
10	84.95	89.05	98.91	98.85
11	83.47	83.47	97.82	97.70
12	82.34	87.29	98.18	98.05
13	81.01	81.02	98.31	98.11
14	79.91	85.83	97.92	97.87
15	79.05	85.54	98.38	98.22
16	77.90	84.42	98.34	98.10
17	76.75	76.75	96.54	96.29
18	76.25	83.15	97.59	97.43
19	75.10	75.08	96.97	96.81
20	74.07	81.59	96.78	96.69
21	72.93	87.32	97.12	97.02
22	71.92	84.74	95.24	95.11
23	71.21	71.23	96.51	96.47
24	70.84	87.31	95.50	95.35
25	74.58	86.76	97.05	97.00
26	73.78	82.96	96.96	96.86
27	72.63	84.72	96.56	96.44
28	72.24	86.13	96.35	96.28
29	71.43	71.43	96.43	96.33
30	70.85	85.76	96.16	96.08
31	69.05	68.98	93.45	93.47
32	68.24	84.63	94.25	94.21
33	68.17	82.16	95.19	95.09
34	67.99	78.40	92.33	92.17
35	66.20	84.25	93.71	93.86
36	66.38	83.86	95.61	95.65
37	64.95	64.93	92.93	93.06
38	65.02	76.22	94.85	94.85
39	63.75	79.61	92.41	92.42
40	63.82	82.40	94.44	94.60
41	62.40	62.42	92.18	92.20
42	62.63	82.97	94.75	94.76
43	61.31	61.26	92.38	92.49
44	61.76	80.38	90.27	90.32
45	60.54	81.16	93.19	93.22
46	59.25	72.21	90.72	91.15
47	59.82	59.80	94.39	94.42
48	58.78	80.71	92.35	92.41
49	57.66	81.01	90.40	90.44
50	58.31	79.93	93.03	93.23
51	57.17	75.24	91.88	92.32
52	56.19	78.16	90.43	90.34
53	55.27	55.28	88.28	88.57
54	55.86	78.86	92.65	92.75
55	54.84	78.27	90.60	91.04
56	53.94	78.25	89.29	89.36
57	53.15	72.72	87.65	87.22

Mighell, K. J. 2010, PASP (in press: October 2010)



[1008.2192v1] CRBLASTER: A Parallel-Processing Computational Framework for Embarrassingly-Parallel Image-Analysis Algorithms

http://arxiv.org/abs/1008.2192v1

arxiv crblaster mighell

[1008.2192v1] CRBLASTER: A Para...

arXiv.org > astro-ph > arXiv:1008.2192v1

Search or Article-id (Help | Advanced search)

All papers Go!

Astrophysics > Instrumentation and Methods for Astrophysics

CRBLASTER: A Parallel-Processing Computational Framework for Embarrassingly-Parallel Image-Analysis Algorithms

Kenneth John Mighell

(Submitted on 12 Aug 2010)

The development of parallel-processing image-analysis codes is generally a challenging task that requires complicated choreography of interprocessor communications. If, however, the image-analysis algorithm is embarrassingly parallel, then the development of a parallel-processing implementation of that algorithm can be a much easier task to accomplish because, by definition, there is little need for communication between the compute processes. I describe the design, implementation, and performance of a parallel-processing image-analysis application, called CRBLASTER, which does cosmic-ray rejection of CCD (charge-coupled device) images using the embarrassingly-parallel L.A.COSMIC algorithm. CRBLASTER is written in C using the high-performance computing industry standard Message Passing Interface (MPI) library. The code has been designed to be used by research scientists who are familiar with C as a parallel-processing computational framework that enables the easy development of parallel-processing image-analysis programs based on embarrassingly-parallel algorithms. The CRBLASTER source code is freely available at the official application website at the National Optical Astronomy Observatory. Removing cosmic rays from a single 800x800 pixel Hubble Space Telescope WFPC2 image takes 44 seconds with the IRAF script lacos_im.cl running on a single core of an Apple Mac Pro computer with two 2.8-GHz quad-core Intel Xeon processors. CRBLASTER is 7.4 times faster processing the same image on a single core on the same machine. Processing the same image with CRBLASTER simultaneously on all 8 cores of the same machine takes 0.875 seconds -- which is a speedup factor of 50.3 times faster than the IRAF script. A detailed analysis is presented of the performance of CRBLASTER using between 1 and 57 processors on a low-power Tilera 700-MHz 64-core TILE64 processor.

Comments: 8 pages, 2 figures, 1 table, accepted for publication in PASP
Subjects: **Instrumentation and Methods for Astrophysics (astro-ph.IM)**
Cite as: arXiv:1008.2192v1 **[astro-ph.IM]**

Download:

- PDF
- PostScript
- Other formats

Current browse context:
astro-ph.IM
< prev | next >
new | recent | 1008

Change to browse by:
astro-ph

References & Citations
- SLAC-SPIRES HEP (refers to | cited by)
- NASA ADS

Bookmark (what is this?)

or . . . just google "crblaster arxiv"

http://www.noao.edu/staff/mighell/crblaster/preprint.pdf

Porting CRBLASTER to the Tile64 processor

(1) Retrieve the compressed source tar ball crblaster.tar.gz:

http://www.noao.edu/staff/mighell/crblaster/crblaster.tar.gz

(2) Uncompress the tar ball:

gunzip crblaster.tar.gz

(3) Unpack the tar ball:

tar -xvf crblaster.tar

(4) Go to the crblaster directory:

cd crblaster

(5) Retrieve the file **Makefile.tilera** :

http://www.noao.edu/staff/mighell/crblaster/Makefile.tilera

Copy it to a file named Makefile in the crblaster directory:

cp Makefile.tilera Makefile

...

http://www.noao.edu/staff/mighell/crblaster/tilera_build.pdf

This work is supported by a grant, Interagency Order No. NNG06EC81I, from the **Applied Information Systems Research (AISR)** Program of the Science Mission Directorate of the National Aeronautics and Space Administration (NASA).