
VisiOmatic

Release 2.0

E.Bertin, C.Marmo, H.Bouy

September 30, 2016

1	Introduction	1
2	Technical overview	3
2.1	Client Side: VisiOmatic	3
2.2	Server-side: iipsrv	3
3	How to use VisiOmatic	5
3.1	The main window	5
3.2	The Coordinates Pane	5
3.3	The Navigation Pane	6
3.4	Advanced Settings	6
4	VisiOmatic set-up and customization	9
5	VisiOmatic programming interface	13
5.1	Classes	13
5.2	Utility functions	27
6	Server installation and configuration	29
6.1	Pre-requisites	29
6.2	Downloading iipsrv	30
6.3	Installing iipsrv	30
6.4	Web server configuration	30
6.5	iipsrv configuration and testing	33
6.6	System configuration	37
7	Acknowledgements	39
	Bibliography	41

Introduction

The purpose of this manual is to help administrators and data scientists setting up and operating **VisiOmatic** and **IIPImage-astro** together.

This manual is divided in four main sections. The first section gives a technical overview of **IIPImage** and **VisiOmatic**, and their capabilities. The second section will guide administrators through installation and configuration of **IIPImage-astro**. The third section explains how to set up and customize the **VisiOmatic** web interface through concrete examples. Finally, the last section provides a full description of the web client **API**.

Technical overview

VisiOmatic and **IIPImage-astro** form a complete remote visualization system for large multiband astronomical image data. The **VisiOmatic** client interface runs in standard web browsers, generating image requests to a server on behalf of the user. These **HTTP** requests are processed in real-time by the **IIPImage-astro** server to compute and deliver **JPEG**-compressed images which are then updated almost immediately in the interface.

2.1 Client Side: VisiOmatic

The **VisiOmatic** web client interface [1] is built on top of the **Leaflet** Javascript mini-framework (e.g., **MapBox**, **OpenStreetMap**, ...). The **VisiOmatic** interface is fully asynchronous, and is particularly immune to connection glitches. It is embeddable in regular Web pages, blog posts, portals, or wiki entries. It is compatible with touchscreen interfaces such as those offered by iOS and Android mobile devices. The position and appearance of widgets is fully customizable through module options and **Cascading Style Sheets**. The graphic engine relies purely on Javascript and **HTML5** and not on proprietary technology such as **Adobe Flash** or **Microsoft Silverlight**. It is fully compatible with the current breed of popular web browsers, including **Mozilla Firefox** (v4.0 and above), **Google Chrome** (v10.0 and above), **Apple Safari** (v5.1 and above), **Microsoft Edge** (v20.10240 and above) and **Opera** (v10.5 and above).

2.2 Server-side: iipsrv

The **IIPImage-astro** server software consists of a modified version of the open-source **IIPImage** package [6][1], an **FCGI** (**Fast Common Gateway Interface**) application written in C++. **IIPImage** operates as a web-service that encodes and streams in real-time large high resolution images which are delivered in the form of compressed “tiles”. It is designed to be fast and bandwidth-efficient with low processor and memory requirements. It is licensed under version 3 of the **GNU General Public License**.

Compared to other existing solutions, **IIPImage** has the enormous advantage of providing on-the-fly **JPEG** compression as well as access to uncompressed pixel data. This means that it can operate directly on science-grade multispectral data stored in floating-point format, and perform operations such as rescaling or filtering before sending out the resulting image to the client. Also, contrary to other solutions dealing with multispectral data, most of the image processing and compositing is done server-side. This dramatically decreases the quantity of information that has to be sent to the browser, and the amount of computing which must be performed client-side, making the exploration of large datacubes from a smartphone a comfortable experience, even through a 3G connection.

2.2.1 Data management

All the data for a given image data cube are stored in a single, huge **BigTIFF** file. The initial conversion from the **FITS** astronomical image format to **BigTIFF** is handled by the **STIFF** software [7]. On modern hardware, the current **STIFF** conversion rate for transcoding a **FITS** file to an **IIPImage**-ready tiled pyramidal **BigTIFF**

ranges from about 5Mpixel/s to 25Mpixel/s (20-100MB/s) depending on the chosen TIFF compression scheme and system I/O performance. Hence in principle FITS frames with dimensions of up to 16k×16k can be converted in a matter of seconds, and just-in-time conversion could be a viable option for such images.

Vector data are stored in [GeoJSON](#) format. GeoJSON offers a compact, yet human-friendly representation of features such as markers, lines, polygons, and can easily be generated from e.g., [CSV](#) or [ds9](#) region files.

2.2.2 Security

Data security is a major issue for online services. In **IIPImage**, three mechanisms allow the webmaster to control the dissemination of data:

- Paths where the data files are located are not accessible from the web; they are only accessible by the FCGI application on the server, and absolute or relative access through upper levels of the directory tree is forbidden.
- The [same origin security policy](#) in web browsers restricts data access to servers located in the same domain as the server hosting the client code, and prevents Javascript applications on websites from other domains sending [AJAX](#) requests directly to the **IIPImage** server. If however data sharing with other domains is a desired feature, **IIPImage** implements the [CORS](#) mechanism to allow such requests from selected or all external websites.
- As other tile-based image servers, **IIPImage** itself would not prevent a user to download a complete high resolution JPEG image through small pieces, using a dedicated script. If this is an issue, **IIPImage** offers a configurable on-the-fly, randomized [digital watermarking](#) feature.

2.2.3 Performance

IIPImage is known for being particularly efficient. Since 2013 a significant amount of work has gone into optimizing further the C++ server code, by streamlining the processing of vectorized floating-point data and improving I/O performance with Terabyte-sized datasets [\[1\]](#). Current server code is able to serve from about 50 to 500 256×256 JPEG tiles per second per CPU core (2.5GHz). The output image stream from a single 16-core server under heavy load (tens of thousands of tile requests per second) is thus capable of saturating a 1 Gbit/s connection.

Provided that the client interface restricts the number of operations possible with the data, caching image tiles is an efficient way of increasing the throughput of the system. In addition to the caching of image tiles done client-side in the browser, **IIPImage** offers two explicit levels of tile caches server-side:





- In memory, integrated within the application
- Distributed, thanks to the [MemCached](#) caching system.

How to use VisiOmatic

3.1 The main window

The Figure below shows the main window of the VisiOmatic web interface in its default configuration.

The main window contains a “slippy map” carrying the current image layer and optional vector overlays, plus a series of widgets. One navigates through the image and its overlays by “dragging” the map to the desired position. On computers this is done by clicking and holding the left button while moving the mouse, or by using the keyboard arrow keys. On touch devices one must press and move a finger throughout the screen.

At the top left, two magnifier buttons can be used to zoom in () or out (). One can also zoom using the mouse wheel or the $-/+$ keys on computers, or with a [pinch gesture](#) on touch devices. The user can switch to/from full screen mode by clicking the  button (third from the top). The  button (last from the top) opens the Advanced Settings Menu.

The coordinates of the *center* of the current field-of-view (indicated by the cross-shaped reticle) are displayed at the top-right of the main window, in the *Coordinate Pane*. In some configurations, a drop-down list allows one to switch between equatorial (RA,Dec) and other types of coordinates.

Below the coordinates, a *Navigation Pane* may be offered, offering a larger view of the current image, as well as the current field of view, represented by a shaded orange box.

The scale, as measured at the center of the main window, is displayed in the lower left corner.

3.2 The Coordinates Pane

The Coordinates Pane allows the user to:

1. check the central coordinates of the current field of view
2. pan to specific coordinates or to a given object.

To move to specific coordinates or objects, simply click in the coordinates pane, and enter the desired coordinates or object name. VisiOmatic uses [Simbad](#) to parse the input coordinates and object names. According to the [Simbad Query-by-Coordinates webpage](#), the following coordinates writings are allowed:

- 20 54 05.689 +37 01 17.38
- 10:12:45.3-45:17:50
- 15h17m-11d10m
- 15h17+89d15
- 275d11m15.6954s+17d59m59.876s
- 12.34567h-17.87654d
- 350.123456d-17.33333d

- 350.123456 -17.33333







while the dictionary of nomenclature for object identifiers can be found [here](#).

3.3 The Navigation Pane

The Navigation Pane allows the user to quickly move over the image. Just drag the shaded orange rectangle representing the current field of view to the desired location. The Navigation Pane can be minimized by clicking on the pair of arrows that appear on the top right when hovering the area.

3.4 Advanced Settings

The Advanced Settings button gives access to a taskbar with five tabs, from top to bottom and as illustrated in *figmix*:

-  Channel Mixing, which allows the user to choose which image channels to use for display or color compositing
-  Image Preferences, which gives the user control over the contrast, color saturation, gamma correction and JPEG compression level. There is also a switch for inverting the color map.
-  Catalog Overlays for superimposing multiple catalogs in vector form, e.g., the 2MASS Point Source Catalog [2] or the SDSS Photometric Catalog [3].
-  Region Overlays, for overlaying Points Of Interest (such as local catalogs) or any local vector data sets in GeoJSON format.
-  Profile Overlays, for plotting image profiles and (pseudo-)spectral energy distributions from the full precision pixel values stored on the server.
-  Documentation, which opens a panel where any web page can be embedded, e.g., an online-manual.

3.4.1 Channel Mixing

The Channel Mixing panel has two modes, which can be selected using the radio buttons located at the top of the dialog: the Single Channel (monochromatic) mode, and the Multi-Channel (color composite) mixing mode.

In Single Channel mode, one can:

- Select an image channel for display
- Select a color map among a selection of four
- Set the minimum and maximum channel levels

In Multi-Channel mode, one can:

- Select an image channel to be included in the color mix
- Set the color this channel contributes to the mix
- Set the minimum and maximum channel levels
- Click on a channel name in the active channel list to edit a channel contributing to the current mix, or click on the trashcan button to remove it.

3.4.2 Image Preferences

The Image Preference dialog gives access to global image display settings:

- Color map inversion (negative mode)
- Contrast (scaling factor)
- Color saturation (0 for black&white, >1 for exaggerating colors)
- Gamma correction (2.2 for linear output on a properly calibrated monitor, higher values for brightening dark regions, lower values for darkening)
- JPEG quality percentage. The lower the quality percentage, the more compressed the image and the more artifacts in the rendering.

Note: Users with a low bandwidth can improve the reactivity of the display by setting a lower JPEG quality percentage.

3.4.3 Catalog Overlays

The Catalog Overlay dialog allows the user to download and overlay catalogs in the current field of view. The available list of catalogs and the rendering of catalog sources (marker, cross, circle with magnitude-dependent radius, ellipse, etc.) depends on client settings. In the current default interface all catalogs are queried from the *VizieR* service at *CDS* [5].

To query a catalog, move the map and adjust the zoom level to the desired field of view; choose the catalog from the drop-down catalog selector and an overlay color from the color selector, then click the “GO” button. After a few seconds a new overlay with the chosen color should appear, as well as a new entry in the active catalog list below the drop down selector. Each overlay may be turned off or on by clicking on the check-mark in the corresponding entry of the active catalog list, or simply discarded by clicking on the trashcan button.

Depending on the implementation in the client, sources may be clickable and have pop-up information windows attached.

3.4.4 Region Overlays

The Region Overlay dialog allows the user to overlay Points/Regions Of Interest, such as local catalogs, detector footprints, etc. These POIs/ROIs may be clickable and have information attached for display in pop-up windows.

The Region Overlay selection mechanism is exactly the same as that of Catalog Overlays.

3.4.5 Profile Overlays



The Profile overlay dialog gives the user the possibility to extract pixel values directly from the scientific data. These data are unaffected by channel mixing, image scaling or compression.

The profile option extracts series of pixel values along lines of constant galactic longitude or latitude. The line color is pink by default and can be changed using the color picker. The line itself is positioned on the image by first dragging the map to the desired start coordinate, pressing the “START” button, and dragging the map to the desired end coordinate and pressing the “END” button. After some calculation, a window appears with a plot of the image profile along the selected line. In mono-channel mode, a single line is plotted that corresponds to the currently selected channel. In color mode, all active channels are plotted, with their channel mixing color. On devices equipped with a mouse one can zoom inside the plots by clicking with the left button and selecting the zoom region. Double-click to zoom out. The plot window can be closed by clicking/touching the small cross in the upper right corner, and reopened at any time by clicking on the line.

The spectrum option, as its name implied extracts pixel data along the channel axis at a given position. The position is selected by dragging the map to the desired coordinate and clicking on the “Go” button. A circle marker appears with the color that was selected using the color picker (purple by default). After some calculation, a window pops out with the “spectrum” of pixel values at the selected coordinate. Note that the ordering of pixel values follows that of the channels in the data cube.

Caution: The absolute flux calibration is valid within the limits described in Section 3.
--

3.4.6 Documentation panel

Clicking on the  symbol (which may be located at the bottom of the taskbar) brings up the online documentation panel. A navigation bar is located at the bottom of the panel to facilitate browsing through the provided documentation or website sections and come back to the main page. Finally, a download button (PDF Symbol ) located at the bottom right of the Manual Pane may be present to allow the user saving an entire manual as a PDF file for offline reading.

VisiOmatic set-up and customization

Let us now describe the HTML and JavaScript code that make up a typical **VisiOmatic**-based web client. We will assume that the **IIPImage-astro** FCGI is already up and ready to serve pixel data on the server (see *Server installation and configuration*).

The **VisiOmatic** web client [1] works essentially as a large **Leaflet** plug-in, providing subproperties to the original **Leaflet** “classes” with new or overloaded methods. The whole client fits in a single HTML file. The relevant part of the HTML file begins with the inclusion of the **Leaflet**, **jqPlot**, **Spectrum** and **VisiOmatic** Cascading Style Sheets in the HTML header:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset='utf-8' name='viewport' content='width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=no'>
  <link rel='stylesheet' href='visiomatic/dist/jqplot.css' />
  <link rel='stylesheet' href='visiomatic/dist/spectrum.css' />
  <link rel='stylesheet' href='Leaflet/dist/leaflet.css' />
  <link rel='stylesheet' href='visiomatic/dist/visiomatic.css' />
```

The Leaflet map itself may be styled (background color, dimensions):

```
<style type='text/css'>
  body {padding: 0; margin: 0; }
  html, body, #map { width: 100%; height: 100%; }
</style>
```

The following Javascript library files must be included: **jQuery** (V2.1.4+), **jqPlot** (V1.0.8+), **Spectrum** (V1.7.1+), **Leaflet** (V1.0) and **VisiOmatic** (V2.0+). Custom Catalog objects may be included too.

```
<script src='visiomatic/dist/jquery-min.js'></script>
<script src='visiomatic/dist/jqplot-min.js'></script>
<script src='visiomatic/dist/spectrum-min.js'></script>
<script src='Leaflet/dist/leaflet.js'></script>
<script src='visiomatic/dist/visiomatic.js'></script>
<script src='catalogs.custom.js'></script>
</head>
```

The whole **Leaflet/VisiOmatic** interface, including advanced menus, fits in a `<div>` HTML element. The element can be styled to occupy the whole HTML body (as in this dedicated viewer example), or a smaller section of an existing web page. There can be several independent viewing areas (“maps”) per web page; each of them must have a different `id` attribute. The map below is given a black background.

```
<body>
  <div id='map' style='background:black'></div>
```

Next comes the JavaScript code. The first thing it does is to capture the URL of the web page. The `L.IIPUtils.parseURL()` utility function (see *description in next chapter*) parses it to generate a dictionary of *query string* keyword/value pairs that we will use to modify specific viewer settings:

```
<script>
  var args = L.IIPUtils.parseURL(window.location.search.substring(1));
```

For instance, if the current webpage URL is `http://myviewer.org?foo=ok&bar=1`, `args['foo']` will contain `'ok'` and `args['bar']` will contain `'1'`.

The `L.map()` command initializes the map inside the HTML element that carries the given id (`'map'` in this example):

```
var map = L.map('map', {fullscreenControl: true, zoom: 5});
```

Note the `fullscreenControl` option (`false` by default) that adds a Full-Screen switch to the regular **Leaflet** interface.

The heart of the viewer code is the instantiation and initialization of the image layer with `L.tileLayer.iip()` (see *next chapter*). `L.tileLayer.iip()` accepts a query string comprised of the FastCGI URL and the name of the data file, plus a set of options:

```
var iip = '/fcgi-bin/iipsrv.fcgi?FIF=cfhtls_d1.ptif',
    layer = L.tileLayer.iip(iip, {
  center: args['center'] ? args['center'] : false,
  fov: args['fov'] ? parseFloat(args['fov']) : false,
  mixingMode: args['mode'] ? args['mode'] : 'color',
  defaultChannel: args['channel'] ?
    parseInt(args['channel'], 10) : 2,
  contrast: 0.7,
  gamma: 2.8,
  colorSat: 2.0,
  channelColors: [[0,0,1],[0,1,1],[0,1,0],[1,1,0],[1,0,0]],
  channelLabels: [
    'u band',
    'g band',
    'r band',
    'i band',
    'z band'
  ],
}).addTo(map);
```

The example above illustrates some options including

- using query arguments in the URL of the present web page to set the initial center coordinates, zoom level, mixing mode and monochromatic channel index for the viewer,
- changing the default image contrast, gamma factor and color saturation,
- applying colors to the different channels (`channelColors`),
- labeling every image channel (`channelLabels`).

The rest of the code instantiates controls that will help the user interact with the data. First of all, a map scale in angular units (pixel units are turned off), a central reticle, and the coordinate input/output widget are set up:

```
L.control.scale.wcs({pixels: false}).addTo(map);
L.control.reticle().addTo(map);

var wcsControl = L.control.wcs({
  coordinates: [{label: 'RA,Dec', units: 'HMS'}],
  position: 'topright'
}).addTo(map);
```

Next a “navigation map” is added, using `L.control.extraMap()`. The navigation map is synchronized with the main map. The tile layer object in the navigation map must be distinct from that of the main map, even if it deals with the same data-cube, as here. For displaying the navigation layer we select only channels #2,#3 and #4:

```

var navlayer = L.tileLayer.iip(iip, {
  channelColors: [[0,0,1],[0,1,0],[1,0,0]],
});

var navmap = L.control.extraMap(navlayer, {
  position: 'topright',
  width: 128,
  height: 128,
  zoomLevelOffset: -6,
  nativeCelsys: true,
}).addTo(map);

```

We now initialize a side-bar that will regroup all advanced controls, and set up the channel mixing controls and the image settings controls:

```

var sidebar = L.control.sidebar().addTo(map);

L.control.iip.channel().addTo(sidebar);
L.control.iip.image().addTo(sidebar);

```

The `L.control.iip.catalog()` command adds a catalogue overlay selection menu to the interface. The list of catalogues below is comprised of the 2MASS [2], SDSS [3], and Hudelot et al. [4] catalogues (the latter is defined in `catalogs.custom.js`):

```

L.control.iip.catalog([
  L.Catalog['2MASS'],
  L.Catalog['SDSS'],
  L.Catalog['Hudelot']
]).addTo(sidebar);

```

Regions/Points Of Interest are managed through a very similar interface; using *region objects* in place of *catalog objects*. In our example the objects' content is assigned right in the call to `L.control.iip.region()`:

```

L.control.iip.region(
  [
    {
      url: 'observation_footprint.json',
      name: 'observation footprint',
      description: 'Footprint of the optical observations',
      color: 'blue',
      load: false
    },
    {
      url: 'poi.json',
      name: 'Molinari+ 2011 POIs',
      description: 'Points of Interest discussed in Einstein et al. 2016',
      color: 'orange',
      load: false
    }
  ],
  { nativeCelsys: true }
).addTo(sidebar);

```

`L.control.iip.profile()` adds a measurement tab:

```

L.control.iip.profile().addTo(sidebar);

```

Finally, `L.control.iip.doc()` can be invoked to add an online documentation tab. Any website may be displayed in the documentation panel; it is sand-boxed inside an HTML `iframe` element. Additionally, a URL to a PDF file may be provided, making a “Download PDF” button pop up in the interface.

```

sidebar.addTabList();
L.control.iip.doc('doc/index.html', {
  pdflink: 'doc/doc.pdf'
});

```

```
    }).addTo(sidebar);  
</script>  
</body>
```

Note the call to `sidebar.addTabList()` which moves the next tab to the bottom of the side bar.

VisiOmatic programming interface

The **VisiOmatic** client provides the following sub-properties to the original Leaflet “classes” with new or overloaded methods:

Class	Purpose
<i>L.TileLayer.IIP</i>	Manage IIP tile layers
<i>L.CRS.WCS</i>	Handle celestial coordinates
<i>L.Control.IIP</i>	Common class for managing interface menus and widgets (not used directly)
<i>L.Control.IIP.Channel</i>	Channel mixing interface
<i>L.Control.IIP.Image</i>	Image settings interface
<i>L.Control.IIP.Catalog</i>	Catalog overlay interface
<i>L.Control.IIP.Profile</i>	Image profile overlay interface
<i>L.Control.IIP.Region</i>	Region- and Point-of-interest interface
<i>L.Control.IIP.Doc</i>	Documentation interface
<i>L.Control.Sidebar</i>	Side menu bar and panels
<i>L.Control.WCS</i>	Coordinate input/output interface
<i>L.Control.Scale.WCS</i>	Celestial or pixel scale line
<i>L.Control.Reticle</i>	Reticle at the center of the map
<i>L.Control.ExtraMap</i>	Secondary map synchronized to the main map
<i>L.RGB</i>	Class for managing R,G,B color triplets that describe color pixels

VisiOmatic also provides set of utility functions, grouped in the *L.IIPUtils* object:

Function	Purpose
<i>L.IIPUtils.requestURL()</i>	Make an Ajax call to the specified server
<i>L.IIPUtils.parseURL()</i>	Parse the given URL and generate a dictionary of query keyword/value pairs
<i>L.IIPUtils.checkDomain()</i>	Return the domain of the given URL
<i>L.IIPUtils.isExternal()</i>	Check if the given URL is from an external domain

5.1 Classes

5.1.1 *L.TileLayer.IIP*

L.TileLayer.IIP manages tile layers with image data queried from an IIPimage tile server.

Usage example

```
var map = L.map('map', {fullscreenControl: true});
var iip = '/fcgi-bin/iipsrv.fcgi?FIF=image.ptif';
var ima = L.tileLayer.iip(iip, {cmap: 'jet'}).addTo(map);
```

Creation

```
L.tileLayer.iip (<String> url, <tilelayer-options> options?)
```

instantiates an IIP tile layer object given a URL `url`.

Options

The constructor supports [all options](#) from the regular `TileLayer` plus

Option	Type	Default	Description
crs	<i>L.CRS.WCS</i> object	Extracted from the data header if available; raw pixel coordinates otherwise	Coordinate Reference or World Coordinate System
nativeCelSys	Boolean	false	True if native coordinates (e.g., galactic coordinates) are to be used instead of equatorial coordinates
center	String	false	World coordinates (either in RA,Dec decimal form or in hh:mm:ss.s±dd:mm:ss.s sexagesimal format), or any <i>Sesame</i> -compliant identifier defining the initial centering of the map upon layer initialization. Sexagesimal coordinates and identifier strings are sent to the <i>Sesame</i> resolver service for conversion to decimal coordinates. Assume x,y pixel coordinates if WCS information is missing. Use false for default map centering.
fov	Float	false	Field of view covered by the map upon later initialization, in world coordinates (degrees, or pixel coordinates if WCS information is missing). Use false for default map zooming.
contrast	Float	1.0	Contrast factor
colorSat	Float	1.0	Color saturation for multi-channel data (0.0: B&W, >1.0: enhance)
gamma	Float	1.0 for 8 or 16 bit images or 2.2 for 32 bit (integer or floating-point) images	Display gamma
cMap	String	'grey'	Colormap for single channels or channel combinations. Valid colormaps are 'grey', 'jet', 'cold' and 'hot'
invertCMap	Boolean	false	Invert Colormap or color mix (like a negative)
quality	Integer	90	JPEG encoding quality in percent
mixingMode	Strings	'color'	Channel mixing mode. Valid modes are 'mono' (single-channel) and 'color'
ChannelColors	Array of <i>L.RGB</i> color triplets	[rgb(0.0,0.0,1.0), rgb(0.0,1.0,0.0), rgb(1.0,0.0,0.0), rgb(0.0,0.0,0.0), ...]	RGB contribution of each channel to the mixing matrix
ChannelLabels	Array of strings	['Channel #1', 'Channel #2', ...]	Channel labels
ChannelUnits	Array of strings	['ADUs', 'ADUs', ...]	Channel units
MinMaxValues	Array of [Float,Float]	Extracted from the data header if available; [[0.0,255.0], [0.0,255.0], ...] otherwise	Pairs of lower,higher clipping limits for every channel
defaultChannel	Integer	0	Default active channel index (used, e.g., in mono-channel mode)
commandString	String	null	Query string for overriding settings during layer initialization.

Public methods

L.TileLayer.IIP provides all the regular *L.TileLayer* methods plus

Method	Returns	Description
<code>getIIPMetaData (<String> url)</code>	String	Send an OBJ request to the server specified by url and return a string with the max-size, tile-size, resolution-number, bits-per-channel, min-max-sample-values and subject IIP meta-data
<code>rgbToMix (<Integer> chan, <L.RGB> rgb)</code>	—	Update the channel mixing matrix according to the rgb contribution of channel chan
<code>updateMono ()</code>	—	Set the layer in monochromatic mode using the current active channel
<code>updateMix ()</code>	—	Update the mixing matrix in color mode using the current ChannelColor and colSat settings

5.1.2 L.CRS.WCS

L.CRS.WCS is a new class that extends the original *L.CRS* object to support celestial projections described by the FITS WCS standard. Currently only the PIXEL (Cartesian), CAR, COE, TAN and ZEA projections are supported, in equatorial, ecliptic, galactic, and supergalactic coordinates.

Usage example

```
var wcs = L.CRS.wcs(header);
```

Creation

`L.CRS.wcs (<String> hdr, options?)`

instantiates a CRS WCS object given a FITS header stored in the `hdr` string.

Options

Option	Type	Default	Description
<code>ctype</code>	{x: String, y: String}	{x: 'PIXEL', y: 'PIXEL'}	WCS projection type for both axes
<code>nativeCelsSys</code>	Boolean	false	If true native coordinates (e.g., galactic coordinates) are to be used instead of equatorial coordinates
<code>naxis</code>	<i>L.Point</i>	[256, 256]	Dimensions of the full resolution image in pixels
<code>nzoom</code>	Integer	9	Number of zoom levels in the pyramid
<code>crpix</code>	<i>L.Point</i>	[129, 129]	Pixel coordinates of the projection center
<code>crval</code>	<i>L.LatLng</i>	[0.0, 0.0]	World coordinates of the projection center
<code>cd</code>	[[Float,Float], [Float,Float]]	[[1.0, 0.0], [0.0, 1.0]]	Jacobian matrix of the de-projection at the projection center
<code>natpole</code>	<i>L.LatLng</i>	[90.0, 180.0]	World coordinates of the native pole

Public methods

L.CRS.WCS provides all the regular *L.CRS* methods plus

Method	Returns	Description
<code>pixelScale (<Float> zoom, <L.LatLng> latLng)</code>	Float	Return the angular pixel scale in degrees at current coordinates and zoom level
<code>rawPixelScale (<L.LatLng> latLng)</code>	Float	Return the angular pixel scale in degrees at current coordinates and at full resolution
<code>fovToZoom (<L.Map> map, <Float> fov, <L.LatLng> latLng)</code>	Float	Return the zoom level that corresponds to the specified Field-of-View in degrees
<code>celsysToEq (<L.LatLng> latLng)</code>	<i>L.LatLng</i>	Convert native celestial coordinates to equatorial coordinates
<code>eqToCelsys (<L.LatLng> latLng)</code>	<i>L.LatLng</i>	Convert equatorial coordinates to native celestial coordinates

5.1.3 L.Control.IIP.Channel

L.Control.IIP.Channel is a new control class for managing the channel mixing interface.

Usage example

```
var control = L.control.iip.channel({cMap: 'jet'}).addTo(map);
```

Creation

`L.control.iip.channel (<control-options> options?)`

instantiates a channel mixing control object in the given mode ('mono' or 'color').

Options

The constructor supports all options from other *L.Control* classes plus

Option	Type	Default	Description
<code>title</code>	String	'Channel mixing'	Title of the dialog window or panel
<code>mixingMode</code>	Strings	null	Channel mixing mode at start. Valid modes are 'mono' (single-channel), 'color', or null for layer settings
<code>cMap</code>	String	'grey'	Colormap applied to the layer when the interface is attached to the map. Valid colormaps include 'grey', 'jet', 'cold' or 'hot'

Public methods

L.Control.IIP.Channel provides all the existing *L.Control* methods plus

Method	Returns	Description
<code>saveSettings (<L.TileLayer.IIP> layer, <String> mode)</code>	—	Save current interface settings for the given layer and mode
<code>loadSettings (<L.TileLayer.IIP> layer, <String> mode)</code>	—	Load current interface settings for the given IIP layer and mode

5.1.4 L.Control.IIP.Image

L.Control.IIP.Image is a new control class for managing the image settings interface.

Usage example

```
var control = L.control.iip.image().addTo(map);
```

Creation

`L.control.iip.image(<control-options> options?)`

instantiates an image control object.

Options

The constructor supports all options from other `L.Control` classes plus

Option	Type	Default	Description
<code>title</code>	String	'Image preferences'	Title of the dialog window or panel

5.1.5 L.Control.IIP.Catalog

L.Control.IIP.Catalog is a new control class for managing the interface that controls catalog queries and catalog overlays.

Usage example

```
var control = L.control.iip.catalog([L.Catalog['2MASS'], L.Catalog['SDSS'],  
L.Catalog['Hudelot']]).addTo(map);
```

Creation

`L.control.iip.catalog(<Array of catalogs>, <control-options> options?)`

instantiates a catalog control object with the given list of catalogs.

Catalogs

Catalog objects describe the catalogs that can be accessed through a pull-down menu in the interface. They have the following properties (there are no default values):

Option	Type	Example	Description
name	String	'My catalog'	Catalog label as it will appear in the pull-down menu
attribution	String	'Catalog of my preferred sources by Me and al. (2015)'	Attribution or credit line
url	String	L.Catalog.vizierURL + '/asu-tsv?&-mime=csv&-source=II/246&-out=2MASS,RAJ2000,DEJ2000,Jmag,Hmag,Kmag&-out.meta=&-c.eq={sys}&-c={lng},{lat}&-c.bd={dlng},{dlat}&-out.max={nmax}'	URL template for the catalog query
color	String	'#FC04A0'	Default overlay color for this catalog
maglim	Float	20.0	Expected limiting magnitude of the catalog (used to scale symbols).
service	String	'Vizier@CDS'	Label describing the web service that provides the catalog
regionType	String	'box'	Angular query type: 'box' or 'cone'
properties	Array of strings	['J', 'H', 'K']	labels for the source properties returned by the catalog service (except coordinates).
units	Array of strings	['mag', 'mag', 'mag']	units of the source properties returned by the catalog service.
objurl	String	L.Catalog.vizierURL + '/VizieR-5?-source=II/246&-c={ra},{dec},eq=J2000&-c.rs=0.01'	URL template for querying individual sources (e.g., when clicking the source ID in the popup source dialog)

Options

The constructor supports all options from other `L.Control` classes plus

Option	Type	Default	Description
title	String	'Catalog overlay'	Title of the dialog window or panel
nativeCelsSys	Boolean	false	True if native coordinates (e.g., galactic coordinates) are to be used instead of equatorial coordinates
color	String	'#FFFF00'	Default catalog overlay color
timeOut	Float	30	Time out delay for catalog queries (in seconds)

5.1.6 L.Control.IIP.Profile

`L.Control.IIP.Profile` is a new control class for managing the interface that controls profile extraction and profile overlays.

Usage example

```
var control = L.control.iip.profile().addTo(map);
```

Creation

`L.control.iip.profile(<control-options> options?)`

instantiates a profile control object.

Options

The constructor supports all options from other `L.Control` classes plus

Option	Type	Default	Description
<code>title</code>	String	'Catalog overlay'	Title of the dialog window or panel
<code>color</code>	String	'#FFFF00'	Default profile overlay color

5.1.7 L.Control.IIP.Region

`L.Control.IIP.Region` is a new control class for managing the region interface.

Usage example

```
var control = L.control.iip.region([
  {
    name: 'Region 1',
    description: 'A first region',
    url: 'region1.json',
    color: 'blue',
    load: false
  }, {
    name: 'Region 2',
    description: 'A second region',
    url: 'region2.json',
    color: 'orange',
    load: false
  }
], {
  nativeCelsys: true
}).addTo(map);
```

Regions are defined in GeoJSON files. Here is an example of the content of a GeoJSON file:

```
{ "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [266.46398, -27.94870], [267.43523, -28.46585],
            [266.47605, -29.85244], [265.49663, -29.32837], [266.46398, -27.94870]
          ]
        ]
      },
      "properties": {
        "description": "Limits of region 1."
      }
    }
  ]
}
```



```

    }
  }, {
    "type": "Feature",
    "geometry": {
      "type": "Point",
      "coordinates": [266.46042, -28.82444]
    },
    "properties": {
      "description": "<h2>A Point of Interest</h2><p>This object should be worth monitoring.</p>"
    }
  }
]
}

```

The description string inside the GeoJSON properties object can be used to provide the HTML content of a window that pops up when the user clicks on the accompanying region. Note that links in the description should preferably be targeted to appear in a new browser window/tab using the `target='blank'` attribute setting.

Creation

`L.Control.iip.region (<Array of regions> regions, <control-options> options?)`

instantiates a region control object given an array of region objects.

Regions

Region objects describe the regions that can be accessed through a pull-down menu in the interface. They have the following properties (there are no default values):

Option	Type	Example	Description
name	String	'My region'	Region label as it will appear in the pull-down menu
description	String	'This is region A'	HTML string to appear in a popup window when clicking over the region (overridden by the <code>description</code> property in the GeoJSON data).
url	String	'myregions.json'	URL of the GeoJSON data
color	String	'#C0AC23'	Default overlay color for this region
load	Boolean	false	If true, region data are automatically loaded as the control is attached to the map.

Options

The constructor supports all options from other `L.Control` classes plus

Option	Type	Default	Description
title	String	'Channel mixing'	Title of the dialog window or panel
nativeCelsSys	Boolean	false	True if native coordinates (e.g., galactic coordinates) are to be used instead of equatorial coordinates
color	String	'#00FFFF'	Default region overlay color
timeOut	Float	30	Time out delay for region downloads (in seconds)

5.1.8 L.Control.IIP.Doc

L.Control.IIP.Doc is a new control class for displaying online documentation.

Usage example

```
var control = L.control.iip.doc('mydoc/index.html', {pdflink: 'mydoc/mydoc.pdf'}).addTo(map);
```

Creation

`L.control.iip.doc (<String> url, <control-options> options?)`

instantiates a doc control object, given a link to the documentation homepage.

Options

The constructor supports all options from other `L.Control` classes plus

Option	Type	Default	Description
title	String	'Documentation'	Title of the dialog window or panel
pdflink	String	'mydoc.pdf'	Link a PDF version of the documentation

5.1.9 L.Control.Sidebar

L.Control.Sidebar is a control class for adding sliding panes to the left or the right of the map. It is derived from the `leaflet-sidebar` plugin by Tobias Bieniek.

Usage example

```
var sidebar = L.control.sidebar().addTo(map);
```

Creation

`L.control.sidebar (<control-options> options?)`

instantiates a sidebar control object.

Options

The constructor supports all options from other `L.Control` classes plus

Option	Type	Default	Description
title	String	'Toggle advanced menu'	String which will appear in the tooltip while hovering over the sidebar toggle button.
forceSeparateButton	Boolean	false	If true, the sidebar toggle button will be separated from the zooming control panel

Public methods

L.Control.Sidebar provides all the existing *L.Control* methods plus

Method	Returns	Description
<code>addTabList ()</code>	<code>tabList</code>	Create and return a new <code>tabList</code> (collection of tabs)
<code>addTab (<String> id, <string> className, <String> title, <Element> content, <String> sideClass)</code>	<code>Element</code>	Add a new tab to the current <code>tabList</code> , inserting content into the associated pane, and return the pane <code>Element</code>
<code>open (<String> id)</code>	—	Open sidebar (if necessary) and show the tab with the specified ID
<code>close ()</code>	—	Close sidebar (if necessary)
<code>toggle ()</code>	—	Collapse or expand the sidebar

5.1.10 L.Control.WCS

L.Control.WCS is a new control class for managing an input/output coordinate widget.

Usage example

```
var control = L.control.wcs(
  {coordinates: [
    {label: 'RA,Dec', units: 'HMS'},
    {label: 'Gal l,b', units: 'deg', nativeCelsys: true}
  ]}
).addTo(map);
```

Creation

`L.control.wcs (<control-options> options?)`
instantiates a WCS control object.

Options

The constructor supports all options from other *L.Control* classes plus

Option	Type	Default	Description
<code>title</code>	String	'Scale'	Name of the control (to be used for tooltips)
<code>coordinates</code>	Array of <i>coordinates</i>	[{ label: 'RA, Dec', units: 'HMS', nativeCelsys: false }]	Set of <i>coordinates</i> settings (see below)

Coordinates

Coordinates objects describe the coordinates that can be selected through a pull-down menu to the left of the coordinate entry widget. They have the following properties (there are no default values):

Option	Type	Example	Description
label	String	'RA, Dec'	Coordinate label as it will appear in the pull-down menu
units	String	'HMS'	Coordinate units/types as they will appear in the interface: 'HMS' for hh:mm:ss.s±dd:mm:ss.s sexagesimal format, 'deg' for decimal degrees, 'other' for raw decimal output.
nativeCelSys	Boolean	false	True if native coordinates (e.g., galactic coordinates) are to be used instead of equatorial coordinates

5.1.11 L.Control.Scale.WCS

L.Control.Scale.WCS is a new control class derived from *L.Control.Scale* that adds a scale to the map. It supports both angular and pixel units.

Usage example

```
var control = L.control.scale.wcs({pixels: false}).addTo(map);
```

Creation

L.control.scale.wcs (<control-scale-options> options?)
 instantiates a WCS scale control object.

Options

The constructor supports all the *L.Control.Scale* options plus

Option	Type	Default	Description
title	String	'Scale'	Name of the control (to be used for tooltips)
degrees	Boolean	true	Whether to show the degree scale line (deg/arcmin/arcsec/mas)
pixels	Boolean	true	Whether to show the pixel scale line
custom	Boolean	false	Whether to show the custom scale line
customScale	float	1.0	Custom scale factor in pixels
customUnits	String	"	Name of the custom scale unit
planetRadius	float	6378137.0	Planet radius in meters (for metric and imperial units)

5.1.12 L.Control.Reticle

L.Control.Reticle is a new control class that adds a crosshair at the center of the map window.

Usage example

```
var control = L.control.reticle().addTo(map);
```

Creation

L.control.reticle ()
 instantiates a reticle control object.

Options

The constructor has no option.

5.1.13 L.Control.ExtraMap

L.Control.ExtraMap is a new control class for displaying an additional map synchronized with the main map, picture-in-picture style. It is an adaptation of [Leaflet-MiniMap](#) by Norkart.

Usage example

```
var control = L.control.extraMap(  
  L.tileLayer.iip('/fcgi-bin/iipsrv.fcgi?FIF=image.ptif'),  
  {  
    position: 'topright',  
    width: 192,  
    height: 128,  
    zoomLevelOffset: -6,  
  }  
) .addTo (map);
```

Creation

`L.control.extraMap (<L.TileLayer> layer, <control-options> options?)`

instantiates an `extraMap` control object with the given layer. Note that the layer must not be shared with the main map or another `extraMap`.

Options

The constructor supports all options from other `L.Control` classes plus

Option	Type	Default	Description
<code>title</code>	String	'Navigation mini-map. Grab to navigate'	Name of the control (to be used for tooltips)
<code>toggleDisplay</code>	Boolean	true	Whether the extraMap should display a minimization button
<code>strings</code>	Array of strings	{hideText: 'Hide map', showText: 'Show map' }	Labels for the toggle button (appear in tooltips)
<code>autoToggleDisplay</code>	Boolean	false	Whether the extraMap should hide automatically if the main map bounds do not fit within the extraMap bounds (useful when <code>zoomLevelFixed</code> is set)
<code>zoomLevelFixed</code>	Boolean or Integer	false	Valid, fixed zoom level for the extraMap, or false if a dynamic <code>zoomLevelOffset</code> is to be applied instead
<code>zoomLevelOffset</code>	Integer	-5	Offset applied to the zoom in the extraMap with respect to that of the main map
<code>zoomAnimation</code>	Boolean	false	Whether the extraMap should have an animated zoom (if true, will cause the extraMap to lag a bit after the main map)
<code>width</code>	Integer	150	Width of the ExtraMap in pixels
<code>height</code>	Integer	150	Height of the ExtraMap in pixels
<code>collapsedWidth</code>	Integer	24	Width of the toggleMarker and the ExtraMap when collapsed, in pixels
<code>collapsedHeight</code>	Integer	24	Height of the toggleMarker and the ExtraMap when collapsed, in pixels
<code>aimingRectOptions</code>	<code>path-options</code>	{color: '#FF7800', weight: 1, clickable: false}	Style of the aiming rectangle (clickable is always forced to false)
<code>shadowRectOptions</code>	<code>path-options</code>	{color: '#803C00', weight: 1, opacity: 0, fillOpacity: 0, clickable: false}	Style of the shadow aiming rectangle (clickable is always forced to false)

Public methods

`L.Control.ExtraMap` provides all the existing `L.Control` methods plus

Method	Returns	Description
<code>changeLayer (<L.TileLayer> layer)</code>	—	Replace the current extraMap layer with the one provided

5.1.14 L.RGB

`L.RGB` is a new class for managing R,G,B color triplets that describe color pixels.

Usage example

```
var rgb = L.rgb(0.2, 0.4, 0.0);
var rgb = L.rgb([0.2, 0.4, 0.0]);
var rgb = L.rgb('#336600');
```

Creation

`L.rgb (<Float> r, <Float> g, <Float> b)` instantiates an RGB object with the given red, green and blue components.

5.2 Utility functions

5.2.1 L.IIPUtils.requestURL()

`L.IIPUtils.requestURL()` is used throughout **VisiOmatic** to send *Ajax* requests to a given URL.

Usage

`L.IIPUtils.requestURL (<String> url, <String> purpose, <Function> action, <Object> context, <Float> timeOut)`

Usage example

```
L.IIPUtils.requestURL(
  'http://cdsweb.u-strasbg.fr/cgi-bin/nph-sesame/-oI/A?M31',
  'getting coordinates for M31',
  function (context, httpRequest) { console.log(context, httpRequest); },
  this,
  10
);
```

Arguments

Name	Type	Description
url	String	URL to which the request must be sent
purpose	String	Short description of the request purpose
action	Function (context, httpRequest)	Reference to a function to which the context and the httpRequest objects will be passed upon completion of the request
context	Object	The context object (e.g., this), will be passed as a first argument to the action function
timeOut	Float	Time out delay in seconds. No time out if argument is missing.

5.2.2 L.IIPUtils.parseURL()

`L.IIPUtils.parseURL()` parses a URL and returns a dictionary of *query string* keyword/value pairs.

Usage

`L.IIPUtils.parseURL (<String> url)`

Usage example

```
args = L.IIPUtils.parseURL(  
    'http://myviewer.org/?channel=2&mode=mono'  
);  
console.log(args['channel'], args['mode']);
```

Arguments

Name	Type	Description
url	String	URL to be parsed

5.2.3 L.IIPUtils.checkDomain()

L.IIPUtils.checkDomain() parses a URL and returns the [domain name](#) associated with it.

Usage

```
L.IIPUtils.checkDomain(<String> url)
```

Usage example

```
domain = L.IIPUtils.checkDomain('http://myviewer.org/test');
```

Arguments

Name	Type	Description
url	String	URL to be parsed

5.2.4 L.IIPUtils.isExternal()

L.IIPUtils.isExternal() parses a URL and returns `true` if it belongs to the current domain or `false` otherwise.

Usage

```
L.IIPUtils.isExternal(<String> url)
```

Usage example

```
flag = L.IIPUtils.isExternal('http://myviewer.org/doc/');
```

Arguments

Name	Type	Description
url	String	URL to be parsed

Server installation and configuration

IIPImage is a FastCGI application and as such it requires a web server to run.

This section describes how to install and configure the **IIPImage** server from scratch. Further information can be found on the official [IIPImage webpage](#).

6.1 Pre-requisites

6.1.1 Hardware

The **IIPImage** server (**ipsrv**) is meant to run on one or several server machines, with multiple CPU cores. The “astronomy-oriented” version of **ipsrv** is largely vectorized but it is single threaded; multiple CPU cores are taken advantage of through multiple instances spawned by the web server.

ipsrv operates on large data files (image data cubes), which can be as large as several Terabytes. The performance of the `astro` version is often I/O-limited, especially when it comes to latency. It is therefore important that the data files be located on a fast storage system, with low access times. Ideally this will be an enterprise-level array of SSDs. The lower performance of slower devices, such as spinning disks, can partially be compensated with random access memory by taking advantage of the operating system caching of file operations, as well as the built-in **ipsrv** caching mechanisms. In all cases, the more memory available for caching there is on the system, the more responsive the server will be under heavy load.

In what follows, we will assume that the image data reside in the `/raid/array/` directory.

6.1.2 Operating System

This installation guide focuses on Linux. Nevertheless **IIPImage** has been designed to be cross-platform and has been successfully tested on Linux, Sun Solaris, Mac OS X and Windows.

6.1.3 Software

Before starting the installation, one should make sure that “development packages” (coming with header files) of the following libraries have been installed on the server:

- LibFCGI v2.4+
- LibJPEG Turbo v1.2+, or libJPEG
- LibTIFF v4.0+
- zLib v1.2+

Note that **ipsrv** relies on the **BigTIFF** format for managing image data files larger than 2GB. If **BigTIFF** support is not included in the **LibTIFF** packages available for your Linux distribution (e.g., because it is too old), a manual installation of **LibTIFF** v4.0+ may be necessary.

6.2 Downloading `iipsrv`

The source package for this version is available on [GitHub](#). We strongly recommend against installing the master version of `IIPImage` at this stage, as it has not yet been optimized for working with the **VisiOmatic** client.

6.3 Installing `iipsrv`

1. Clone the project:

```
$ git clone https://github.com/cmarmo/iipsrv-astro.git
```

2. Enter the project directory:

```
$ cd iipsrv-astro
```

3. Generate configuration files for compilation and installation:

```
$ sh autogen
$ ./configure
```

- Although the command above should work in most cases, the `configure` script offers many customization options (see `./configure --help`), including the possibility to change the paths where include and library files are located. For instance for managing manual installations of the TIFF library:

```
$ ./configure --with-tiff-includes=<DIR> --with-tiff-libraries=<DIR>
```

- The Intel compiler, **icc**, is able to vectorize loops containing transcendental functions and generally provides superior performance in **iipsrv**, compared to the GNU compiler. If **icc** is installed on your system, the following configuration line will generate an executable optimized for a wide range of machines based on INTEL processors:

```
$ ./configure CXX=icc CXXFLAGS="-O3 -axSSSE3,SSE4.1,SSE4.2,AVX,CORE-AVX2,CORE-AVX-I -no-prec-
```

4. Compile and install:

```
$ make
$ sudo cp src/iipsrv.fcgi /<your>/<fcgi-bin>/<directory>
```

6.4 Web server configuration

The real life performance of **iipsrv** for serving tiles critically depends on the HTTP web server configuration. It depends even more on the HTTP server itself, especially in high concurrency environments.

6.4.1 Assessing server performance

In [1], four HTTP server packages — **Apache** v2.4.1, **lighttpd** v1.4.35, **NGINX** v1.4.7 and **OpenLiteSpeed** v1.2.7 — were benchmarked with **iipsrv**. Fig. 6.1 shows how efficiently each of these servers responds to bursts of tile queries for various levels of concurrency (the number of queries in each burst). Tile queries were simulated using a modified version of the **Apache Benchmarking tool** benchmarking tool on a local machine, connected through a 10GbE link. The result is expressed in terms of throughput (the number of tiles per seconds) and latency (the average time it takes for tiles to be returned). Perfect server behaviour would consist of constant throughput, and latency that increases linearly with concurrency beyond some threshold, as queries are queued inside the server pipeline.

As can be seen, HTTP server packages behave very differently in this test. If we set the limit for acceptable latency to 1 second, we see that a 12-core system is able to manage bursts of up to 2,000 simultaneous queries

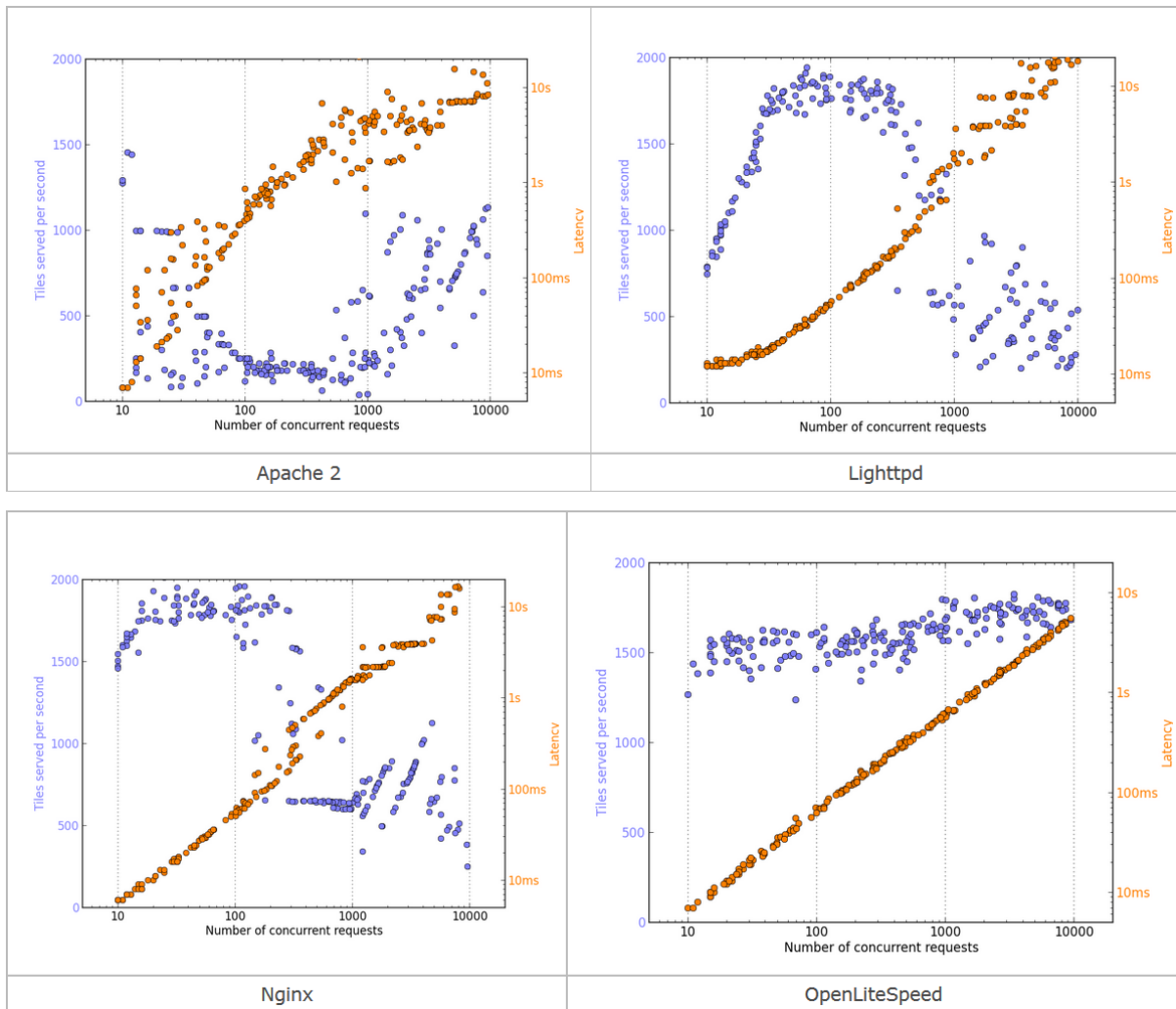


Fig. 6.1: IIPImage-astro tile-serving throughput and latency as a function of concurrency for four different HTTP servers running on the same 12-core Linux system.

with **OpenLiteSpeed**, which corresponds to about 100 users frantically browsing through the image. With **Apache**, these numbers are ten times lower, and both throughput and latency behave much more erratically.

All HTTP server packages were configured so as to maximize tile serving performance. Although some aspects may have not been fully optimized, **OpenLiteSpeed** comes out as the clear winner in this test. This is why we recommend it for running **IIPImage**. Nevertheless, in the sections below we also provide configuration guides for all four packages.

6.4.2 Apache

The `mod_fastcgi` module must be installed and enabled. A directory containing FastCGI programs should be created and readable by **Apache** processes. Make sure that your **Apache** configuration file (or the configuration file for the FCGI module) contains the following lines

```
LoadModule fastcgi_module /path/to/apachemodules/mod_fastcgi.so
# Create a directory for the iipsrv binary
ScriptAlias /fcgi-bin/ "/path/to/fcgi/directory/fcgi-bin/"
#
# Set the options on that directory
<Directory "/path/to/fcgi/directory/fcgi-bin/">
  AllowOverride None
  Options None
  # Syntax for access is different in Apache 2.4 - uncomment appropriate version
  # Apache 2.2
  #   Order allow,deny
  #   Allow from all
  #
  # Apache 2.4
  Require all granted
</Directory>
```

Finally, the **iipsrv** configuration file, `iipsrv.conf` must be copied in the **Apache** configuration directory (e.g., `/etc/httpd/conf.d/`). The following `iipsrv.conf` features typical settings for a 12-core machine:

```
# Set our environment variables for the IIP server
FcgidInitialEnv VERBOSITY "0"
FcgidInitialEnv LOGFILE "/tmp/iipsrv.log"
FcgidInitialEnv MAX_IMAGE_CACHE_SIZE "100"
FcgidInitialEnv JPEG_QUALITY "90"
FcgidInitialEnv MAX_CVT "3000"
FcgidInitialEnv MEMCACHED_SERVERS "localhost"
FcgidInitialEnv FILESYSTEM_PREFIX "/raid/iip/"
# Define the idle timeout as unlimited and the number of # processes we want
FcgidIdleTimeout -1
FcgidMaxProcessesPerClass 12
```

6.4.3 lighttpd

lighttpd comes with built-in FastCGI support. To configure **iipsrv**, add in your **lighttpd** directory an ASCII file `iipsrv.conf` containing

```
fastcgi.server = ( "/fcgi-bin/iipsrv.fcgi" =>
  ( ( "host" => "127.0.0.1",
      "port" => 9000,
      "check-local" => "disable",
      "min-procs" => 1,
      "max-procs" => 12,
      "bin-path" => server_root + "/fcgi-bin/iipsrv.fcgi",
      "bin-environment" => (
        "LOGFILE" => log_root + "/iipsrv.log",
        "VERBOSITY" => "3",
```

```

        "MAX_IMAGE_CACHE_SIZE" => "100",
        "FILENAME_PATTERN" => "_pyr_",
        "JPEG_QUALITY" => "90",
        "MAX_CVT" => "3000",
        "MEMCACHED_SERVERS" => "localhost",
        "FILESYSTEM_PREFIX" => "/raid/iip/"
    )
))
)

```

For best performances edit the **lighttpd** server settings in `/etc/lighttpd/lighttpd.conf`:

```

server.use-ipv6 = "disable"
server.document-root = server_root + "/html"
server.max-fds = 300000
server.stat-cache-engine = "fam"
server.max-connections = 100000
server.max-keep-alive-idle = 4
server.max-keep-alive-requests = 4

```

Then restart the **lighttpd** server.

6.4.4 NGINX

6.4.5 OpenLiteSpeed

The latest stable version of the **OpenLiteSpeed** web server (**lsws**) can be downloaded from <http://open.litespeedtech.com/>. On most machines, installing **lsws** from the source package is as simple as

```

$ ./configure
$ make
$ sudo make install

```

On RedHat-like systems (e.g., RedHat, Fedora, CentOS,...), the **OpenLiteSpeed** web server is started with

```

$ service lsws start

```

OpenLiteSpeed comes with a graphical web interface which is by default accessible on port 7080 of the server. The default administrator login and password are `admin` and `123456`. Any change to the server configuration made in the interface requires applying a “Graceful Restart” (follow instructions on the web page).

Configuring **OpenLiteSpeed** for **iipsrv** starts by adding a new “virtual host”, or modifying the default one that comes with **OpenLiteSpeed** (Fig. 6.2).

The virtual host menu has several tabs. For this FastCGI application we are mostly concerned with the `Basic`, `External App` and `Context` tabs.

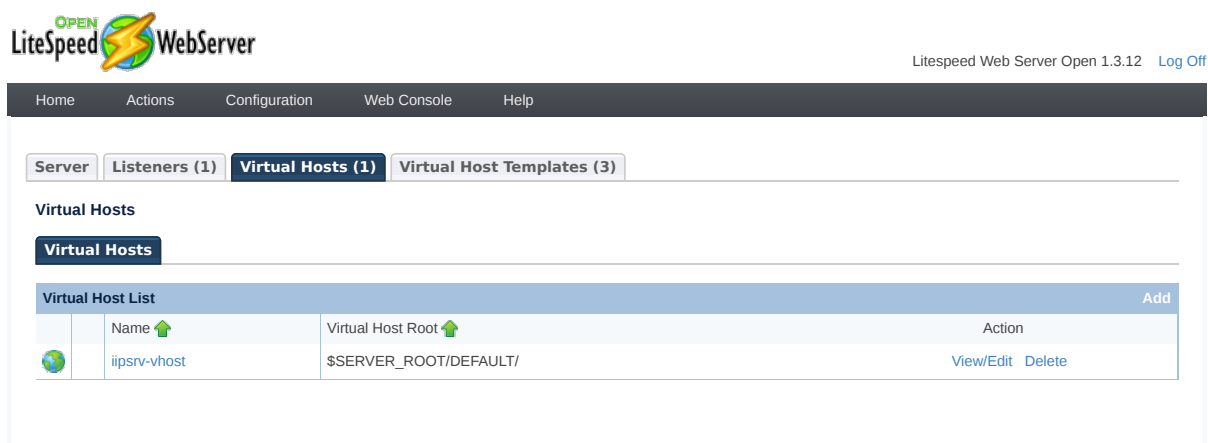
In the `Basic` tab, one should change the `Virtual Host Name` field to, e.g., `iipsrv-vhost`, and set `ExtApp Set UID Mode` to `DocRoot UID` (see Fig. 6.3).

The `External App` tab is where the FastCGI executable must be defined and where it can be fine-tuned. Fig. 6.4 shows an example of a configuration which is appropriate for a 12-core server (`Max Connections = 12` and `Instances = 12`) with high throughput (`Connection Keepalive Timeout = 5`).

Finally, an FCGI entry should be added in the `Context` tab (Fig. 6.5).

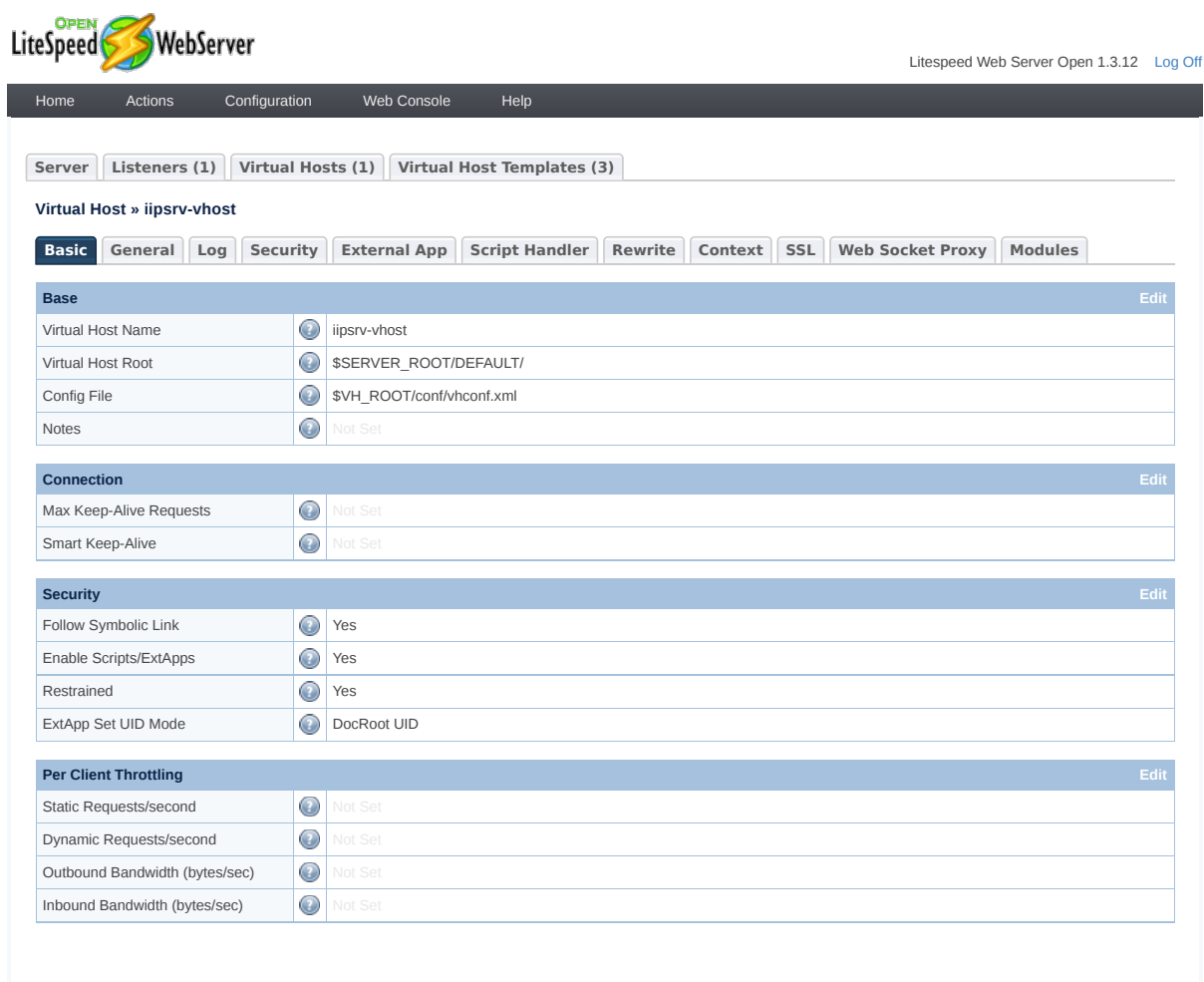
6.5 iipsrv configuration and testing

The [IIPImage webpage](#) gives a description of all the configuration parameters known to **iipsrv**. The following parameters require particular attention:



Copyright © 2013-2015 LiteSpeed Technologies, Inc. All Rights Reserved.

Fig. 6.2: A single “virtual host” is sufficient for operating **iipsrv**.



Copyright © 2013-2015 LiteSpeed Technologies, Inc. All Rights Reserved.

Fig. 6.3: Example of a configuration for the **Basic** tab in the **Virtual Host** section.

Home Actions Configuration Web Console Help

Server Listeners (1) Virtual Hosts (1) Virtual Host Templates (3)

Virtual Host » iipsrv-vhost

Basic General Log Security **External App** Script Handler Rewrite Context SSL Web Socket Proxy Modules

- Give a name that easy to remember, other places will refer to this app by its name.
- Address can be IPv4 socket address "IP:PORT", like 192.168.1.3:7777 or localhost:7777 or Unix domain socket address "UDS://path" like UDS://tmp/lshhttpd/myfcgi.sock.
- UDS is chrooted in chroot environment.
- For local FCGI, Unix domain socket is preferred due to security and better performance. If you have to use IPv4 socket, set the IP part to localhost or 127.0.0.1, thus the FCGI is inaccessible from other machines.
- Local FCGI can be started by the web server. In this case, you need to specify path, backlog and number of instances.

FastCGI App Definition		Save	Back
Name	<input type="text" value="iipsrv-app"/>		
Address	<input type="text" value="127.0.0.1:9000"/>		
Notes	<input type="text" value="IIPImage Fast-CGI server"/>		
Max Connections	<small>number valid range: 1 - 2000</small> <input type="text" value="12"/>		
Environment	<input type="text" value="VERBOSITY=0
JPEG_QUALITY=90
MAX_CVT=3000
MAX_IMAGE_CACHE_SIZE=100
MEMCACHED_SERVERS=localhost"/>		
Initial Request Timeout (secs)	<small>number >= 1</small> <input type="text" value="15"/>		
Retry Timeout (secs)	<small>number >= 0</small> <input type="text" value="15"/>		
Persistent Connection	<input checked="" type="radio"/> Yes <input type="radio"/> No <input type="radio"/> Not Set		
Connection Keepalive Timeout	<small>number valid range: -1 - 10000</small> <input type="text" value="5"/>		
Response Buffering	<input type="text" value="Yes"/>		
Auto Start	<input type="text" value="Yes"/>		
Command	<input type="text" value="/var/www/fcgi-bin/iipsrv.fcgi"/>		
Back Log	<small>number valid range: 1 - 100</small> <input type="text" value=""/>		
Instances	<small>number valid range: 0 - 1000</small> <input type="text" value="12"/>		
suEXEC User	<input type="text" value=""/>		
suEXEC Group	<input type="text" value=""/>		
umask	<input type="text" value=""/>		
Run On Start Up	<input type="text" value="No"/>		
Max Idle Time	<small>number >= -1</small> <input type="text" value=""/>		
Priority	<small>number valid range: -20 - 20</small> <input type="text" value=""/>		

Fig. 6.4: Example of a configuration for the External App tab in the Virtual Host section.

The screenshot shows the LiteSpeed WebServer configuration interface. At the top, there is a navigation bar with 'Home', 'Actions', 'Configuration', 'Web Console', and 'Help'. Below this, there are tabs for 'Server', 'Listeners (1)', 'Virtual Hosts (1)', and 'Virtual Host Templates (3)'. The 'Virtual Hosts (1)' tab is selected, showing 'Virtual Host » iipsrv-vhost'. Underneath, there are sub-tabs: 'Basic', 'General', 'Log', 'Security', 'External App', 'Script Handler', 'Rewrite', 'Context', 'SSL', 'Web Socket Proxy', and 'Modules'. The 'Context' tab is active, displaying a table for 'FCGI Context Definition'. The table has columns for the field name and its value. The 'Notes' field contains the text 'send to iipsrv Fast-CGI'. At the bottom of the page, there is a copyright notice: 'Copyright © 2013-2015 LiteSpeed Technologies, Inc. All Rights Reserved.'

FCGI Context Definition		Edit Delete Back
URI	/cgi-bin/	
Fast CGI App	[VHost Level]: iipsrv-app	
Notes	send to iipsrv Fast-CGI	
Extra Headers	Not Set	
Realm	Not Set	
Authentication Name	Not Set	
Require (Authorized Users/Groups)	Not Set	
Access Allowed	Not Set	
Access Denied	Not Set	
Authorizer	Not Set	
Add Default Charset	Off	
Customized Default Charset	Not Set	
Enable IP GeoLocation	Not Set	

Fig. 6.5: Example of a FastCGI entry in the Context tab of the Virtual Host section.

- `FILESYSTEM_PREFIX` is a prefix added by `iipsrv` to the image data path for all queries. **For security reasons, it is strongly advised** to set `FILESYSTEM_PREFIX` to a path which does not directly or indirectly lead to a system or user directory, or to any file data that must remain inaccessible to the users.
- `VERBOSITY` should be set to 0 for performance reasons.
- `CORS` manages [Cross Origin Resource Sharing](#). It must be set to `*` if the tiles are to be accessible to any client (such as 3rd party applications) outside of those provided by the web server itself.
- `MAX_IMAGE_CACHE_SIZE` sets the size of the **`iipsrv`** JPEG image cache (in MB), which is allocated in memory for every instance. Typical values range from 100 to 2000, depending on the amount of memory in the server, and the number of **`iipsrv`** instances.
- `MEMCACHED_SERVERS` can be used to specify a comma-separated list of IP addresses with optional port numbers (e.g., `127.0.0.1:8888`) that provide caching capabilities using the [Memcached](#) protocol. It is strongly advised to use Memcached in all cases where high traffic loads involving majoritarily identical tile queries are to be expected, e.g., for public outreach applications.

Once the server installed and configured, pointing a web browser to the **`iipsrv`** CGI URL without any argument (e.g., `'http://myurl/cgi-bin/iipsrv.cgi'`) should return a web page similar to that of [Fig. 6.6](#).



Fig. 6.6: Web page returned by the **`iipsrv`** CGI in the absence of arguments.

6.6 System configuration

Server performance also depends on system settings. For maximizing server responsiveness under high concurrency, we recommend following the prescriptions of [\[8\]](#). Some appropriate `sysctl.conf` are given on the [G-WAN website](#) and were used with great success during **`iipsrv`** tests. They are reproduced below:

```
fs.file-max = 300000
net.core.netdev_max_backlog = 400000
net.core.optmem_max = 10000000
net.core.rmem_default = 10000000
net.core.rmem_max = 10000000
net.core.somaxconn = 100000
net.core.wmem_default = 10000000
net.core.wmem_max = 10000000
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.ip_local_port_range = 1024 65535
net.ipv4.tcp_congestion_control = bic
net.ipv4.tcp_ecn = 0
net.ipv4.tcp_max_syn_backlog = 12000
net.ipv4.tcp_max_tw_buckets = 2000000
net.ipv4.tcp_mem = 30000000 30000000 30000000
net.ipv4.tcp_rmem = 30000000 30000000 30000000
net.ipv4.tcp_sack = 1
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_wmem = 30000000 30000000 30000000

# optionally, avoid TIME_WAIT states on localhost no-HTTP Keep-Alive tests:
# "error: connect() failed: Cannot assign requested address (99)"
# On Linux, the 2MSL time is hardcoded to 60 seconds in /include/net/tcp.h:
# #define TCP_TIMEWAIT_LEN (60*HZ)
# The option below is safe to use:
net.ipv4.tcp_tw_reuse = 1

# The option below lets you reduce TIME_WAITs further
# but this option is for benchmarks, NOT for production (NAT issues)
# net.ipv4.tcp_tw_recycle = 1

# Increase nf_conntrack
net.netfilter.nf_conntrack_max = 262144
```

Acknowledgements

VisiOmatic implements services provided by the *Sesame* name resolver and the *VizieR* catalog access tool developed at *CDS*, Strasbourg, France.

Development of VisiOmatic v2.0 was made possible by the ESA/ESTEC Faculty through a grant acquired by G. Pilbratt for the ESAimage viewer project.

- [1] E. Bertin, R. Pillay, and C. Marmo. [Web-based visualization of very large scientific astronomy imagery](#). *Astronomy and Computing*, 10:43–53, 2015.
- [2] R. M. Cutri, M. F. Skrutskie, S. van Dyk, C. A. Beichman, J. M. Carpenter, T. Chester, L. Cambresy, T. Evans, J. Fowler, J. Gizis, E. Howard, J. Huchra, T. Jarrett, E. L. Kopan, J. D. Kirkpatrick, R. M. Light, K. A. Marsh, H. McCallon, S. Schneider, R. Stiening, M. Sykes, M. Weinberg, W. A. Wheaton, S. Wheelock, and N. Zacarias. [VizieR Online Data Catalog: 2MASS All-Sky Catalog of Point Sources \(Cutri+ 2003\)](#). *VizieR Online Data Catalog*, 2246:0, 2003.
- [3] K. N. Abazajian, J. K. Adelman-McCarthy, M. A. Agüeros, S. S. Allam, C. Allende Prieto, D. An, K. S. J. Anderson, S. F. Anderson, J. Annis, N. A. Bahcall, and et al. [The Seventh Data Release of the Sloan Digital Sky Survey](#). *ApJS*, 182:543–558, 2009.
- [4] P. Hudelot, J.-C. Cuillandre, K. Withington, Y. Goranova, H. McCracken, F. Magnard, Y. Mellier, N. Regnault, M. Betoule, H. Aussel, J. J. Kavelaars, P. Fernique, F. Bonnarel, F. Ochsenbein, and O. Ilbert. [The final Canada-France-Hawaii Telescope Legacy Survey Survey \(CFHTLS\) release](#). *VizieR Online Data Catalog*, 2012.
- [5] F. Ochsenbein, P. Bauer, and J. Marcout. [The VizieR database of astronomical catalogues](#). *Astronomy and Astrophysics Supplement Series*, 143:23–32, 2000.
- [6] Denis Pitzalis, Ruven Pillay, and Christian Lahanier. [A new concept in high resolution internet image browsing](#). In *10th International Conference on Electronic Publishing*, 75–85. Banskó, Bulgaria, June 2006. Bulgarian Academy of Sciences, Bulgaria. 10th International Conference on Electronic Publishing, organised by the Bulgarian Academy of Sciences, Bulgaria, 14-16 June 2006.
- [7] E. Bertin. [Displaying Digital Deep Sky Images](#). In P. Ballester, D. Egret, and N. P. F. Lorente, editors, *Astronomical Data Analysis Software and Systems XXI*, volume 461 of Astronomical Society of the Pacific Conference Series, 263. September 2012.
- [8] Bryan Veal and Annie Foong. [Performance scalability of a multi-core web server](#). In *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, ANCS '07, 57–66. New York, NY, USA, 2007. ACM.