Making of ALMAWebQL v2

The National Astronomical Observatory of Japan

Astronomy Data Center
Japanese Virtual Observatory

May 2016

The Developer

Chris Zapart

A research fellow working in the JVO project.

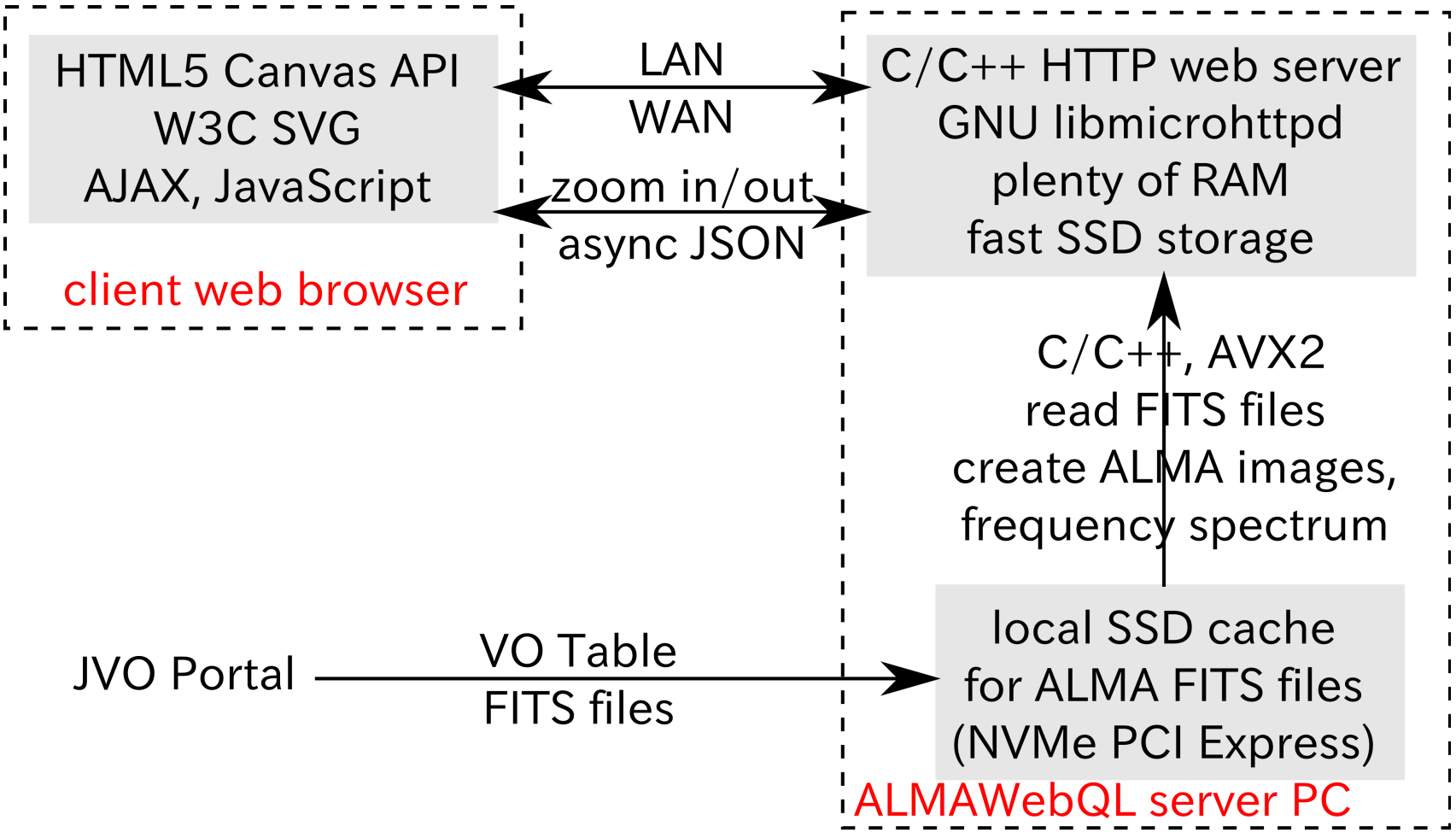
He prepared the presentation !

ALMAWebQL v2

Features:

- an interactive preview of ALMA datasets
- a rich Internet application built on AJAX, HTML5 and SVG
- a custom web server built on top of the GNU libmicrohttpd C library
- real-time image zooming
- near real-time frequency spectrum updates
- partial FITS downloads
- HiDPI display support with automatic image/font rescaling (4K display ready)

Modern client-server architecture



server: NVMe PCI-E SSD

10GB FITS files load in a few seconds

NVMe PCI Express SSD
Intel DC P3600 1.6TBx2, RAID0
ALMAWebQL v2 FITS cache

2.5" SATA3 SSD
1TBx8, RAID5
main FITS storage

under the hood: mmaped FITS files

- custom multi-threaded reading of FITS files from SSD into RAM (cfitsio library too slow with 10GB+ FITS)
- on-the-fly ENDIAN and FLOAT32 to FLOAT16 (half-float) conversion whilst reading FITS
- FITS files mmapped to Linux kernel user space memory instead of read()/write() calls
- efficiency gains: mmap avoids extra memory buffers
- Linux kernel device drivers use mmap too

under the hood: multi-threading

- OpenMP multi-threading (16-core server, dual Intel Xeon E5-2640 v3 @ 2.60GHz CPU)
- Intel C/C++ compiler for the main C code
- efficient AVX2 SIMD vectorisation with the lesser-known Intel SPMD Program Compiler

  `https://ispc.github.io/`
- Intel SPMD compiler accelerates frequency spectrum calculation
- image creation routines also employ Intel SPMD compiler

under the hood: memory allocation

- multi-threaded jemalloc memory allocator
  `http://www.canonware.com/jemalloc/`
- faster than glibc malloc/free
- avoids memory fragmentation when dealing with large data (i.e. arrays > 10GB)
- FITS files stored in a half-float (16-bit FLOAT) little-endian binary format in a custom ALMAWebQL v2 FITS cache
- 50% memory consumption reduction (processing a 24GB FITS file consumes 12GB RAM server-side)

multi-threaded PNG image creation

a custom C code 10x faster than libpng

- parallel lossless PNG zlib compression (NO dependency on slow sequential libpng)
- a single image split into multiple chunks to be compressed separately in parallel zlib streams
- independent adler32 checksum calculation in each zlib stream
- adler32 zlib checksums are combined at the end of the process
- parallel checksumming of PNG IDAT chunks

LZ4-compressed FITS headers

- text compresses very well (typically 5x)
- FITS headers contain a lot of duplicated strings
- compressed FITS headers sent from the server to the web browser as base64-encoded JSON strings
- client-side JavaScript LZ4 decompression (in a browser)

revision v2.1 coming soon... stay tuned

power to the user: colour, “beam” shape selection...