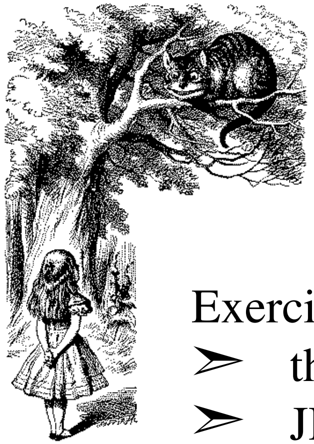
http://aldwww.cern.ch/Jedi



The results

Exercise made easy by:

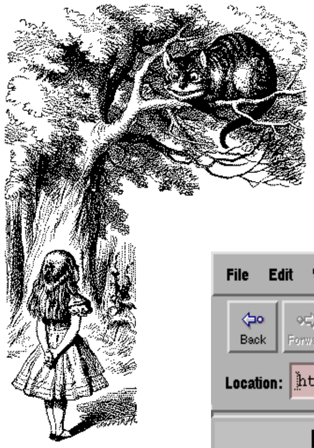
- ➢ the Java language robust, efficient, OO, no "anonymous" pointers
- ➢ JDK (Java Development Toolkit) release 1.1: tools, availability, documentation
- ➢ AWT (Abstract Window Toolkit) support library
- ➢ TCP/IP support library
- ➢ multi-threading support library
- ➢ availability of WWW-based Java interpreters

Problems:

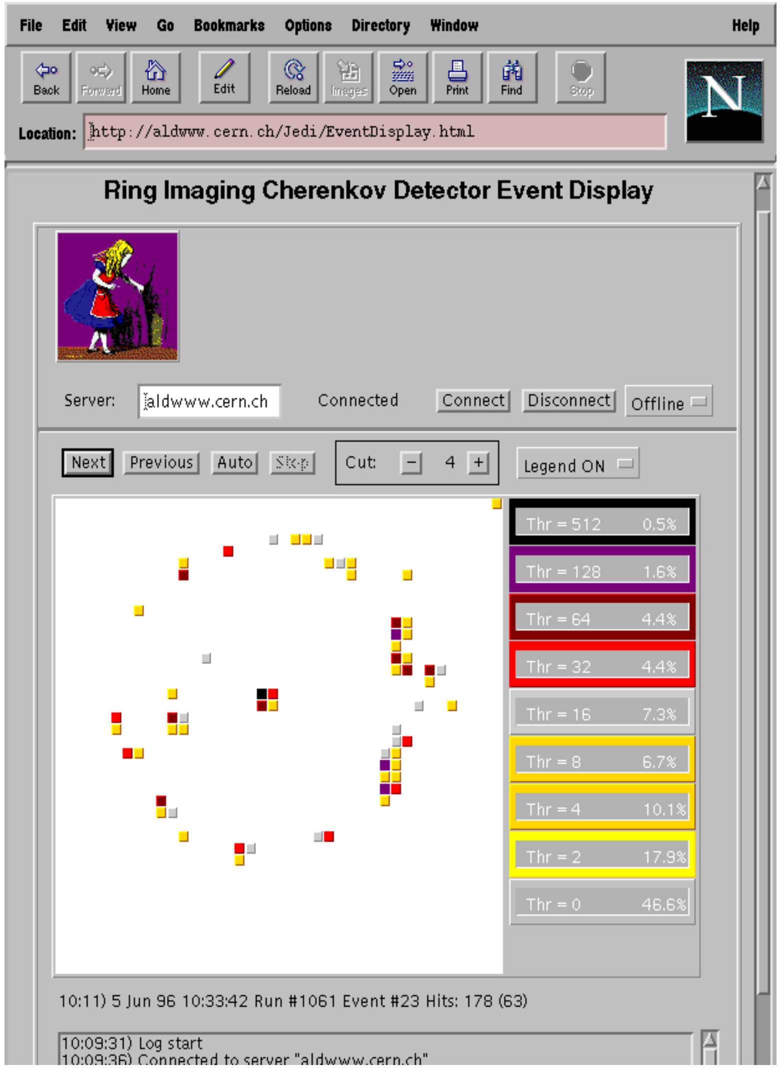
- ♠ debugging can be a nightmare

Result:

- ✓ easy to learn (2 man-week including self training and original code analysis)
- ✓ all platforms validated
- ✓ same functionality on all platforms, in agreement with the native look-and-feel
- ✓ easy to install (both Netscape & JDK)
- ✓ reasonable performances:
  - ☞ ~ 5 seconds to download the Applet + Classes
  - ☞ > 400 KB/s raw data transfer (including optional byte swapping) on ~ 6 KB/Event
  - ☞ 4/6 events/second browsing through a static data file



The Java-based Event Display (JEDI)



Location: http://aldwww.cern.ch/Jedi/EventDisplay.html

Ring Imaging Cherenkov Detector Event Display

Server: aldwww.cern.ch Connected Offline

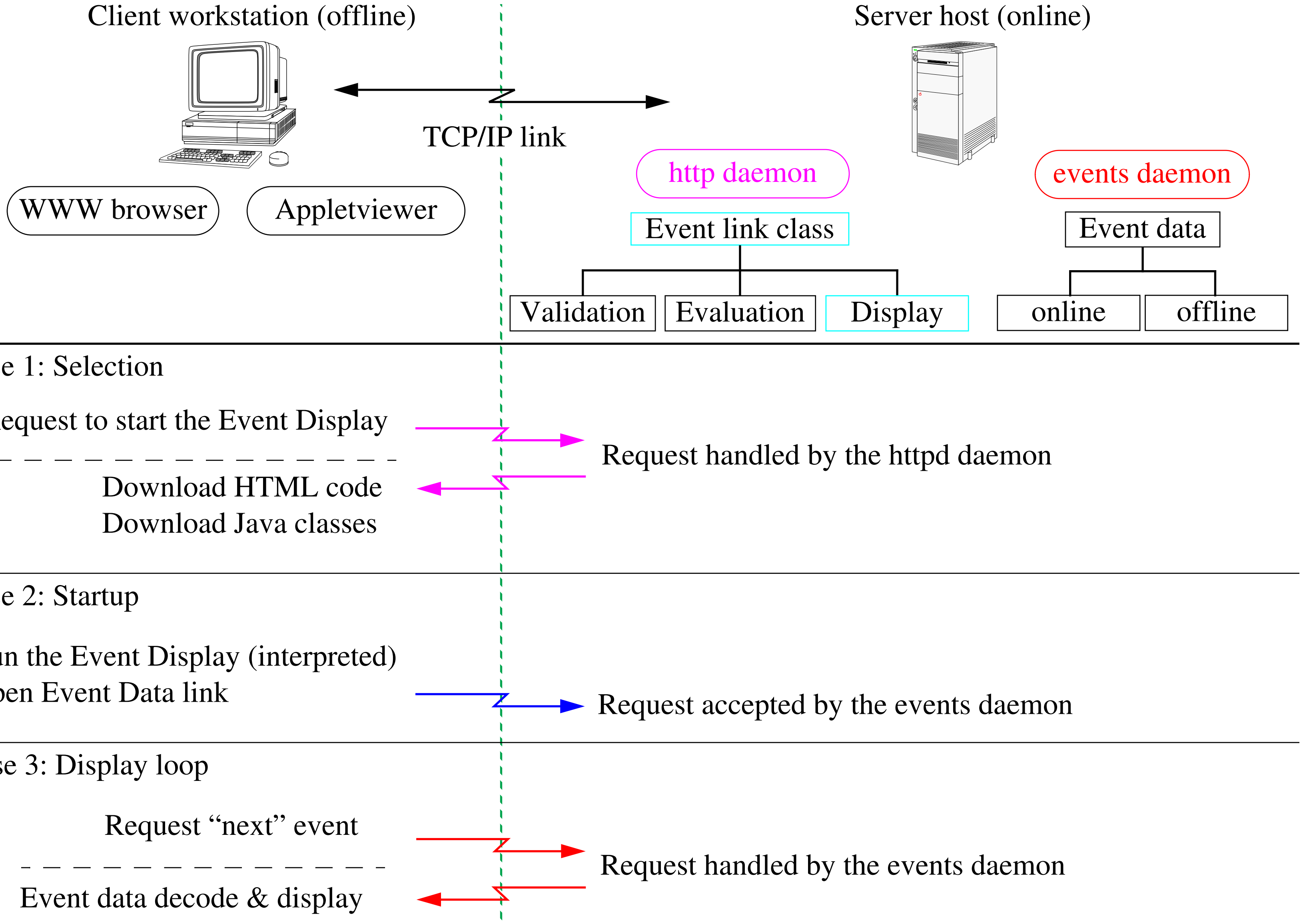
Cut: 4

Thr = 512 0.5%
Thr = 128 1.6%
Thr = 64 4.4%
Thr = 32 4.4%
Thr = 16 7.3%
Thr = 8 6.7%
Thr = 4 10.1%
Thr = 2 17.9%
Thr = 0 46.6%

10:11) 5 Jun 96 10:33:42 Run #1061 Event #23 Hits: 178 (63)

```
10:09:31) Log start
10:09:36) Connected to server "aldwww.cern.ch"
```

The Java-based approach



Client workstation (offline)

Server host (online)

TCP/IP link

WWW browser

Appletviewer

http daemon

Event link class

Validation

Evaluation

Display

events daemon

Event data

online

offline

Phase 1: Selection

Request to start the Event Display

Request handled by the httpd daemon

Download HTML code
Download Java classes

Phase 2: Startup

Run the Event Display (interpreted)
Open Event Data link

Request accepted by the events daemon

Phase 3: Display loop

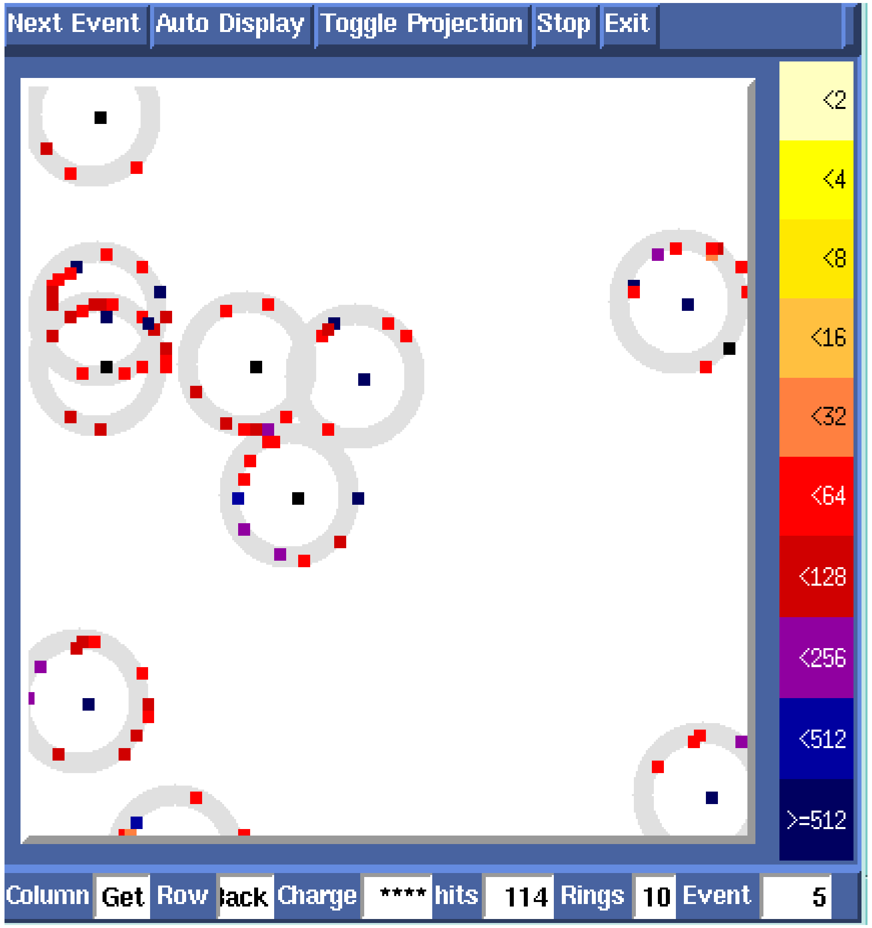
Request “next” event

Request handled by the events daemon

Event data decode & display

The Event Display

We choose an existing application as an evaluation task:



Column Get Row ack Charge **** hits 114 Rings 10 Event 5

Ring Imaging CHerenkov detector event display:

- written in Tcl/Tk
- single-host

We "Javanized" it:

- re-written in Java
- run via WWW httpd server
- client/server model
- dedicated TCP/IP link C ⇨ Java
- added some "extra" features

Target architectures:

- Solaris
- HP-UX
- Windows 95
- AIX
- MAC-OS
- Windows NT

Target browsers:

- Netscape
- Internet Explorer
- JDK appletviewer

Why Java?

Because Java offers a better approach!

The client host:
- ➾ loads the application from the server
- ➾ runs the application locally (interpreter-based)
- ➾ issues requests to the server in application optimized format

The server host:
- ➾ executes the command(s)
- ➾ ships the results in application/network optimized format

The client host:
- ➾ decodes the results
- ➾ displays the results

✓ Better use of the CPU of the client (desktop) host
✓ Less load on the CPU of the server (online) host
✓ Less load on the network
✓ All 🔑 are still satisfied, as long as ∃ a Java interpreter

Why WWW?

WWW-based tools satisfy several of the 🔑 requirements:

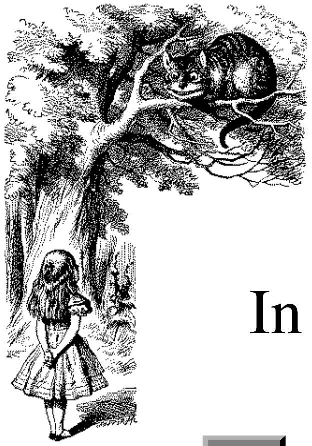
- ☞ based on a client/server approach
- ☞ installed with a single click of the mouse
- ☞ updated automatically when revisited
- ☞ portability guaranteed by availability and popularity of WWW browsers
- ☞ easy to reconfigure and to secure thanks to the httpd daemon builtin features

Problem: load balance between WWW client and server & impact on the network

- the WWW client (desktop):
  - ➾ issues the request
- the WWW server (online):
  - ➾ gets and validates all input parameters
  - ➾ executes the command(s)
  - ➾ handles the results and transform them in a suitable format
  - ➾ ships the results in a format suitable for exchange of hypertext/graphics
- the WWW client (desktop):
  - ➾ displays the results

This will create extra loads on:

- the WWW server: *f(complexity of input phase, format of results)*
- the network: *f(format of results)*



DAQ Systems: the requirements

In the past years, the rules of the game have radically changed

- multi-CPU environment (cooperant & concurrent)
- quick & easy to install and update
- portable on “incompatible” platforms
- low impact on CPU & interconnecting network
- products must be easy to re-configure
- the server (desktop) can be as powerful as the client (online)!

Evaluation of Data Acquisition tools based on Java

Contents:

- ✎ DAQ systems requirements
- ✎ Why WWW?
- ✎ Why Java?
- ✎ The evaluation project
- ✎ The Java-based approach
- ✎ The results