Breaking the Performance Wall: The Case for Distributed Digital Forensics

By

Vassil Roussev, Golden Richard

Presented At

The Digital Forensic Research Conference

DFRWS 2004 USA Baltimore, MD (Aug 11th - 13th)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

http:/dfrws.org

Breaking the Performance Wall: The Case for Distributed Digital Forensics

Golden G. Richard III

Associate Professor, Dept. of Computer Science, University of New Orleans
Technical Advisor, Gulf Coast Computer Forensics Laboratory (GCCFL)
Co-founder, Digital Forensics Solutions

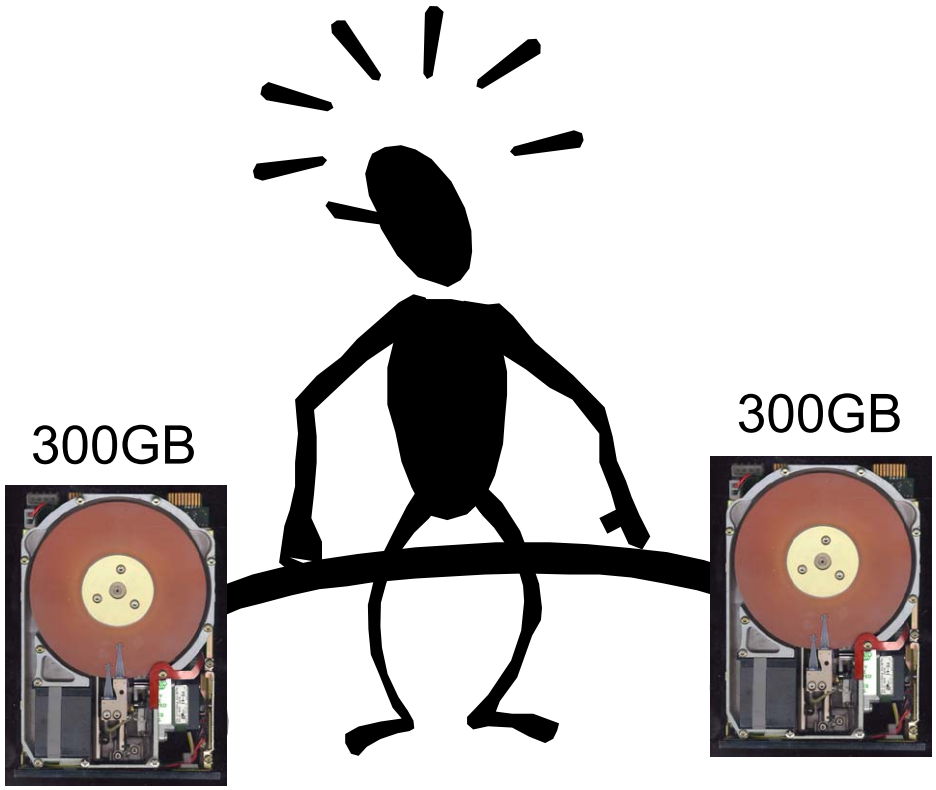
golden@cs.uno.edu

Vassil Roussev

Assistant Professor, Dept. of Computer Science, University of New Orleans

vassil@cs.uno.edu

Single CPU Forensics Investigations



Single CPU State of the Art

- ✓ "Start right away" approach (e.g., Encase)
  - Little preprocessing of evidence
  - Can start working right away…
  - Ridiculously slow searches for large disk images
  - Unacceptable performance for larger images
- ✓ Preprocessing-first approach (e.g., FTK)
  - Substantial preprocessing of evidence
  - Can't work on case until after preprocessing completes
  - …can take days
  - Have keyword indices, thumbnails, etc. available
  - Ridiculously slow searches for un-indexed things
  - Example: regular expression searches
- ✓ Common trait: Too slow.

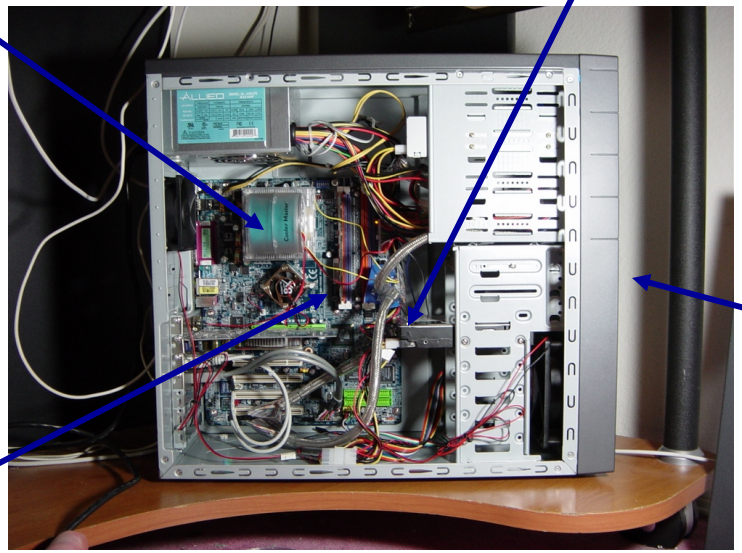
Symptoms

- ✓ Machines tied up for days doing preprocessing
- ✓ Painful to “think outside the box” (i.e., outside the index) during investigation
- ✓ If a regular expression search takes an entire day to complete, what do you do in the meantime?
- ✓ Hint: It requires a $300 video card
- ✓ For large collections of digital evidence, no extra resources to devote to “cooler” tools

The Culprit: The **Entire** Machine

CPU driving investigation... For simple analysis, **OK**. **But it's holding us back.**

Hard drives storing digital evidence for investigation… **Capacity OK**. **WAY too slow.**



Case is **OK**.

Memory. **WAY too small.**

What could we be doing?

- ✓ Summaries for digital video files
  - Extraction of key frames
- ✓ Better image classification
  - Beyond hashing—feature identification
- ✓ Searching audio files for voiceprints
- ✓ Generation of searchable text from audio files using speech recognition
- ✓ Automatic detection of steganography
- ✓ Backgrounding digital evidence preprocessing…
  - Analysis of evidence during preprocessing phase
  - means…investigatory phase can start right away
- ✓ Live searches…regular expression searching…even on huge drive images…uninterrupted brainstorming!
- ✓ What else do you want to do?

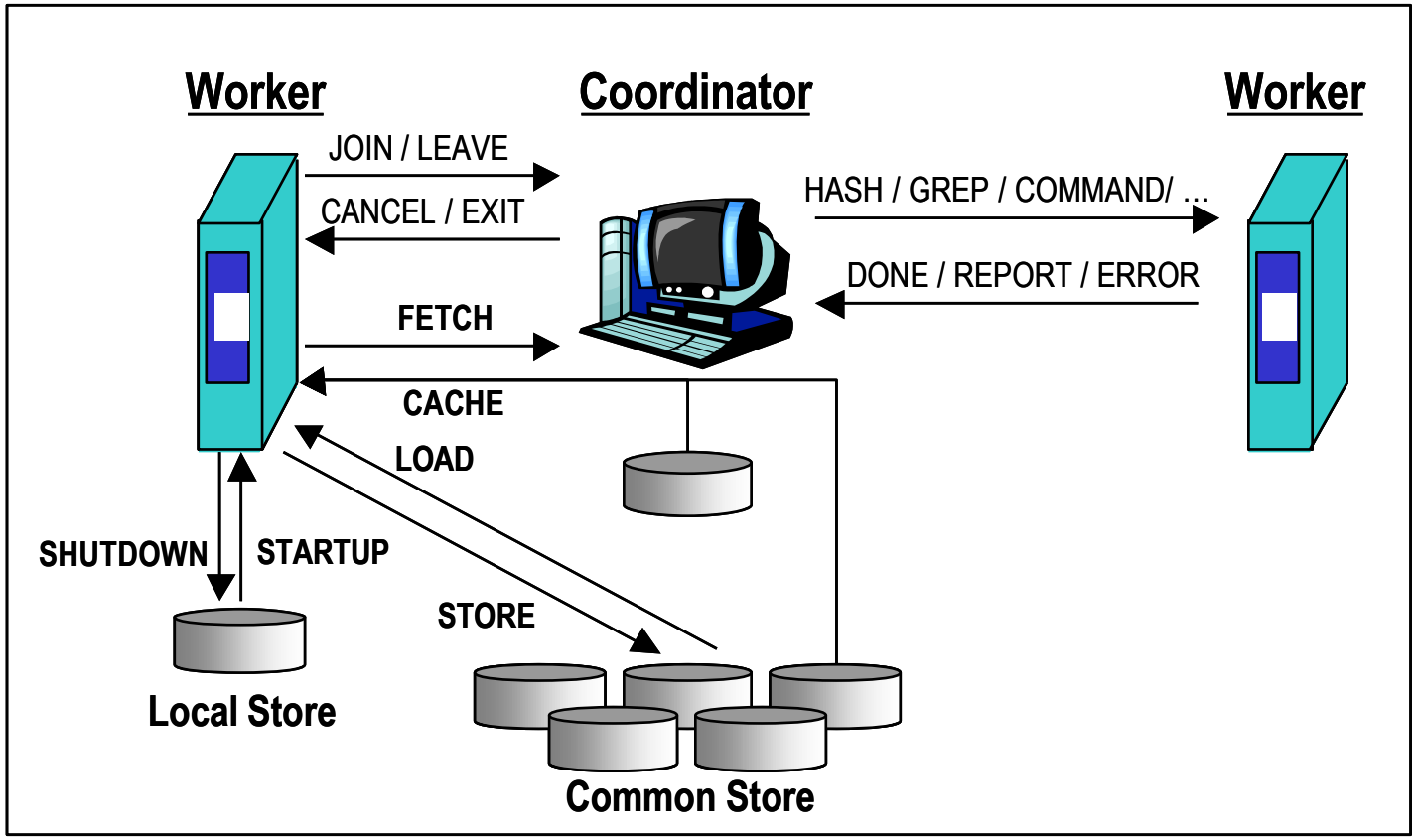
The Solution: Distributed Forensics

- ✓ Forensics analysis on a cluster
- ✓ More CPUs == more horsepower for sophisticated processing
- ✓ If you have high performance file storage, e.g., a RAID…
- ✓ …can load the file server much more effectively
- ✓ More memory == can cache digital evidence for analysis
- ✓ Cache entire disk image in memory of cluster machines

Why don't we already do this?

- ✓ Cluster computing is fairly mature…
- ✓ Why so few distributed forensics tools?
- ✓ Field is young?
- ✓ Requires more sophistication on the part of tool builders?
- ✓ Cost?
  - ~$125K for a dedicated machine that can process 200GB images entirely in memory
  - Can be clever and do it for less
- ✓ We're not being elitist. We just can't take it anymore!

Distributed Digital Forensics (DDF) Framework



Requirements For ~~Framework~~

- ✓ Scalable
  - Want to support at least IMAGE SIZE / RAM_PER_NODE nodes
- ✓ Platform independent
  - Want to be able to incorporate any (reasonable) machine that's available
- ✓ Lightweight
  - Horsepower is for forensics, not the framework—less fat
- ✓ Highly interactive
- ✓ Extensible
  - **Allow incorporation of existing sequential tools**
  - e.g., stegdetect, image thumbnailing, file classification, hashing, …
- ✓ Robust
  - Clusters can be 'notorious'
  - Must handle failed nodes smoothly

Goals for ~~Framework~~

- ✓ Allow investigation to begin “immediately” after drive image is loaded
- ✓ SIGNIFICANTLY speed up traditional evidence processing
- ✓ Beat the hell out of high performance file servers
- ✓ Cache all evidence in RAM
- ✓ Enable new investigatory techniques
- ✓ *N* machines → greater than *N-fold* speedup
- ✓ Brainstorming == ON during investigation. No extensive idle time for human allowed

Protocol

<id> JOIN <name> <cache_size> <IP> <port>
<id> LEAVE <name>
<id> EXIT
<id> SHUTDOWN <name>
<id> STARTUP <name>
<id> CACHE <file_name> <file_size> <hash> <reply_port> <file_data>
<id> FETCH <file_name> <reply_port>
<id> LOAD <files> <reply_port>
<id> STORE <files> <reply_port>
<id> FREE <files> <reply_port>
<id> DONE
<id> ERROR <code> <message>
<id> REPORT <report>
<id> PROGRESS <processed> <all>
<id> CANCEL <req_id>

Protocol (2)

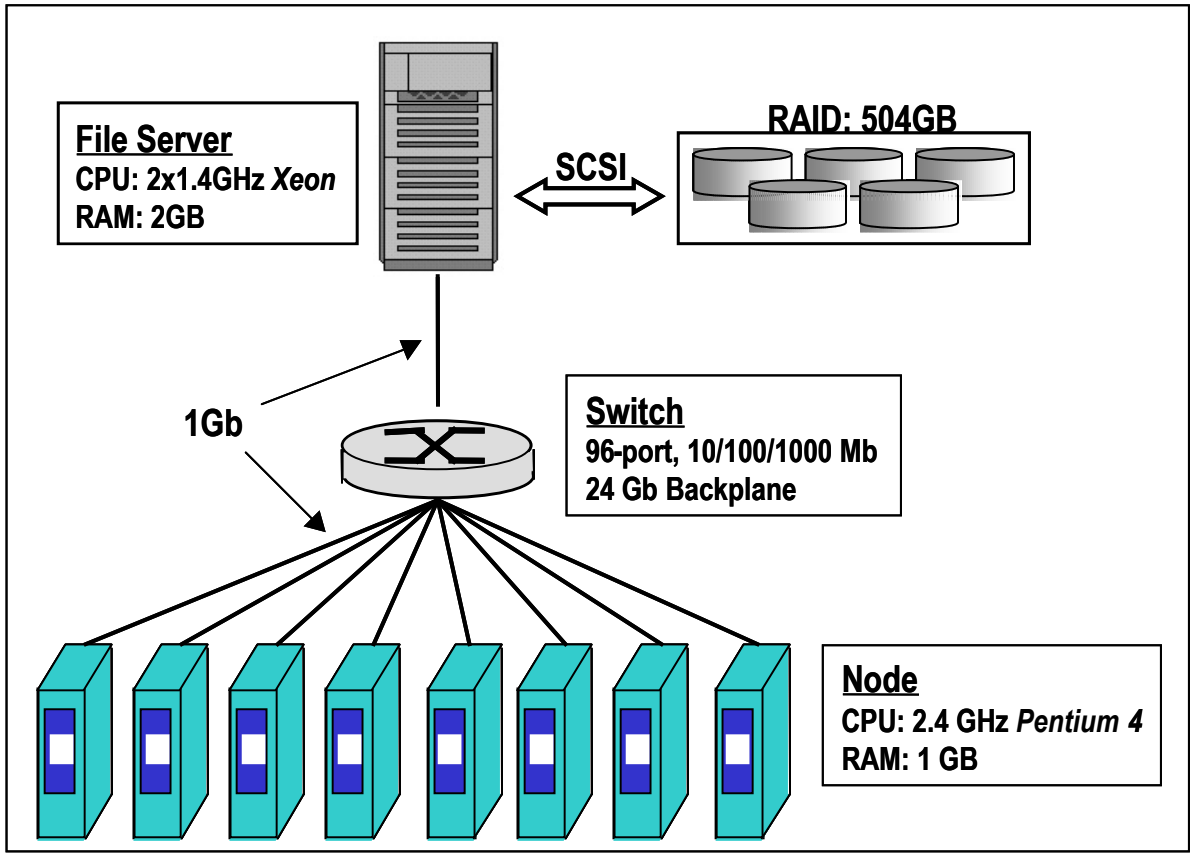
<id> CLASSIFY <files> <progress> <reply_port>
<id> HASH <method> <files> <progress> <reply_port>
<id> GREP <expr> <files> <progress> <reply_port>
<id> THUMB {<files>|<type>} <tdir> <progress> <reply_port>
<id> STEGO {<file>|<type>} <progress> <reply_port>
<id> CRACK <file> <key_range> <progress> <reply_port>
<id> EXEC <command> <arguments> <reply_port>

Experimental Setup

…all your data are belong to us…

72 x 2.4GHz P4s

Experimental Setup (2)



A Few Preliminary Results

- ✓ Target:
  - Dell Optiplex GX1 w/ 6.4GB IDE drive
  - NTFS, ~110,000 files in ~7,800 directories
  - Imaged using dd w/ a Linux boot disk
- ✓ Machine used for "traditional" investigation:
  - 3GHz P4, 2GB RAM, 2 x 73GB 15Krpm Ultra320 SCSI
  - FTK v1.43a

Initial Operation	Time (hh:mm:ss)
FTK Add Evidence	1:38:00
CACHE	0:09:36
8-node LOAD	0:03:58
1-node LOAD	0:05:19

more nodes better at loading the fileserver

Results (2)

- ✓ Live string search:
  "Vassil Roussev"
- ✓ Regular expression search:
  v[a-z]*i[a-z]*a[a-z]*g[a-z]*r[a-z]*a

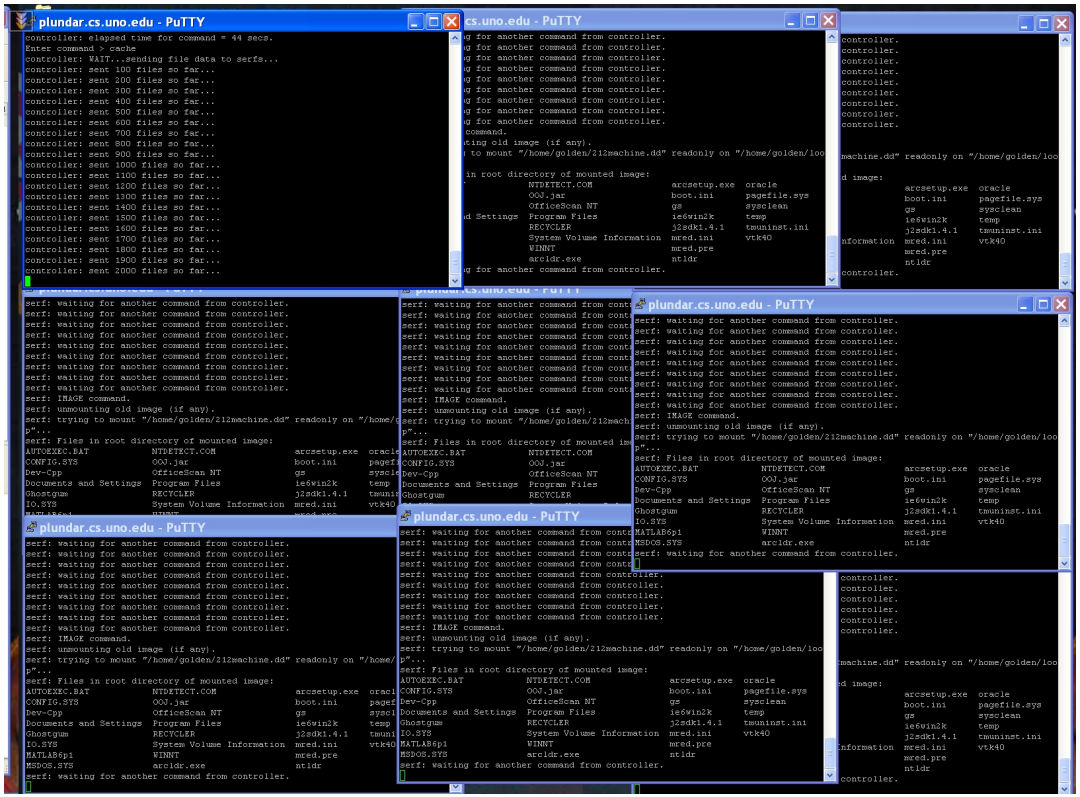
	Search time: String Expression (mm:ss)	Search time: Regular Expression (mm:ss)
FTK	08:27	41:50
8-node System	00:27	00:28

A Different Experiment

- ✓ Stego detection using Stegdetect 0.5 under RH9 Linux on the cluster
- ✓ Traditional:
  - 6GB image mounted using loopback device
  - `find /mnt/loop -exec ./stegdetect '{}' \;`
  - 790 seconds == 13:10 minutes
- ✓ Using the distributed framework
  - Stegdetect 0.5 code incorporated into ~~framework~~
  - Detection against cached files
  - "STEGO" command (after IMAGE/CACHE)
  - 82 seconds == 1:22 minutes
- ✓ 9.6X faster with 8 machines
- ✓ CPU bound operation

To Do…

- ✓ User interface! (unless you love Putty)



To Do (2)

- ✓ Code cleanup
- ✓ Case persistence
- ✓ Better fault tolerance
- ✓ Intelligent caching schemes to support larger images
- ✓ Will swap save us?
- ✓ Collaboration with colleagues (you?) working in:
  - Image analysis/classification
  - Speech recognition
  - Stego
  - Other CPU horsepower-intensive, forensics-applicable stuff

golden@cs.uno.edu

golden@digitalforensicssolutions.com