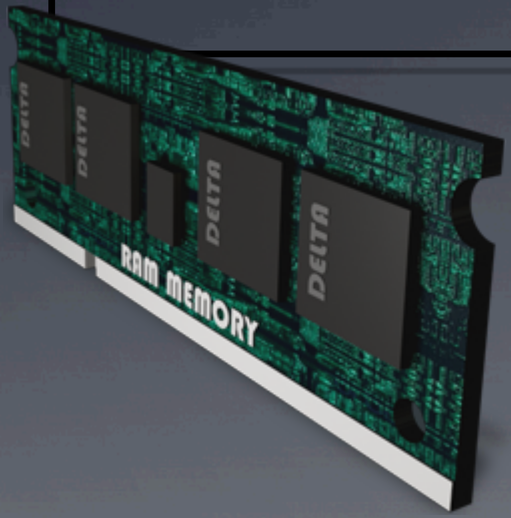
3RD ANNUAL

The Sleuth Kit and
Open Source Digital Forensics Conference

October 2, 2012 TUTORIALS / October 3, 2012 CONFERENCE
WESTFIELDS MARRIOTT, CHANTILLY, VIRGINIA

WWW.OSDFCON.ORG

In RAM We Trust:
A Modern Approach to Forensic Processing

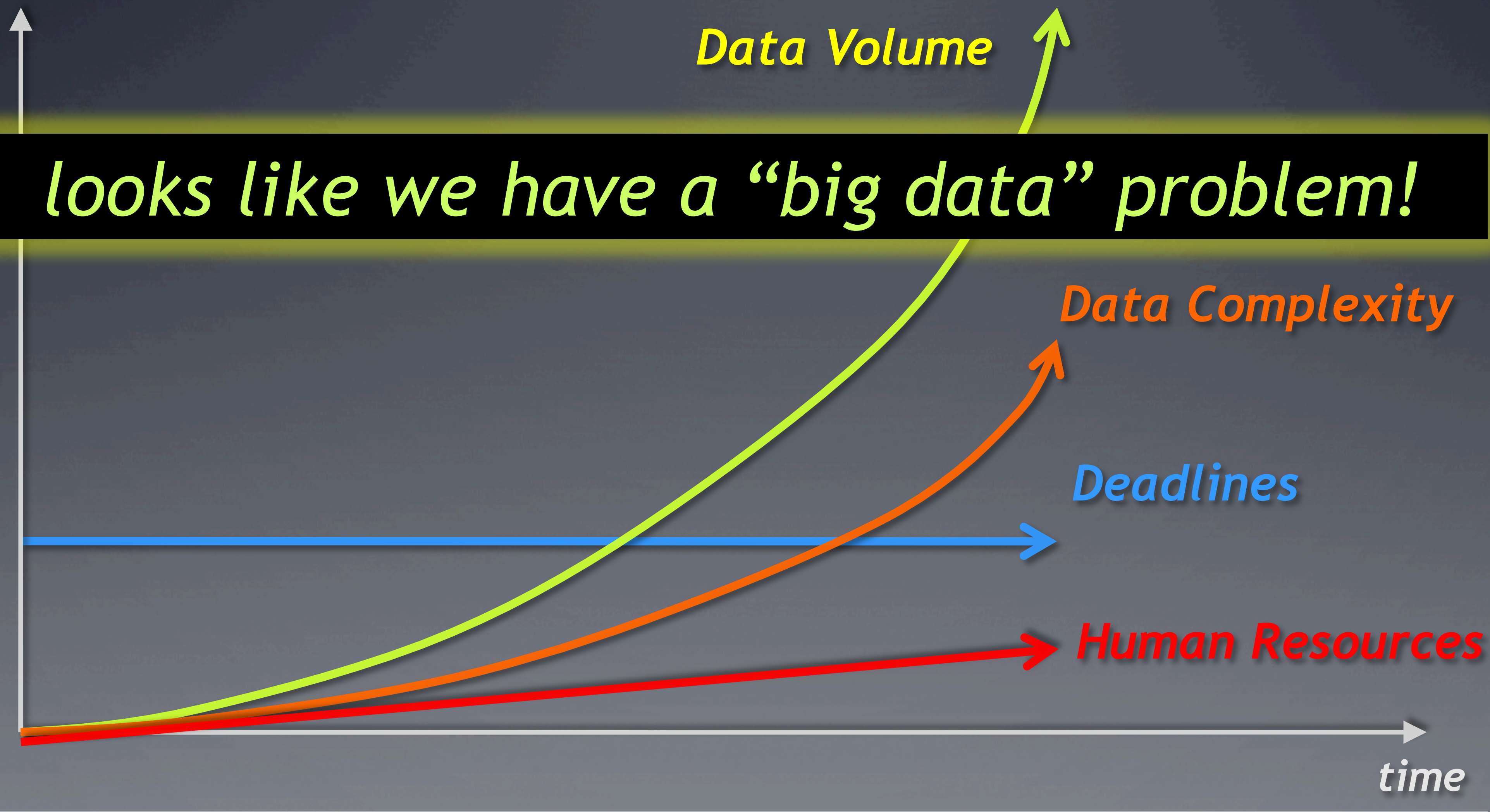


THE UNIVERSITY of NEW ORLEANS

Vassil Roussev

vassil@roussev.net

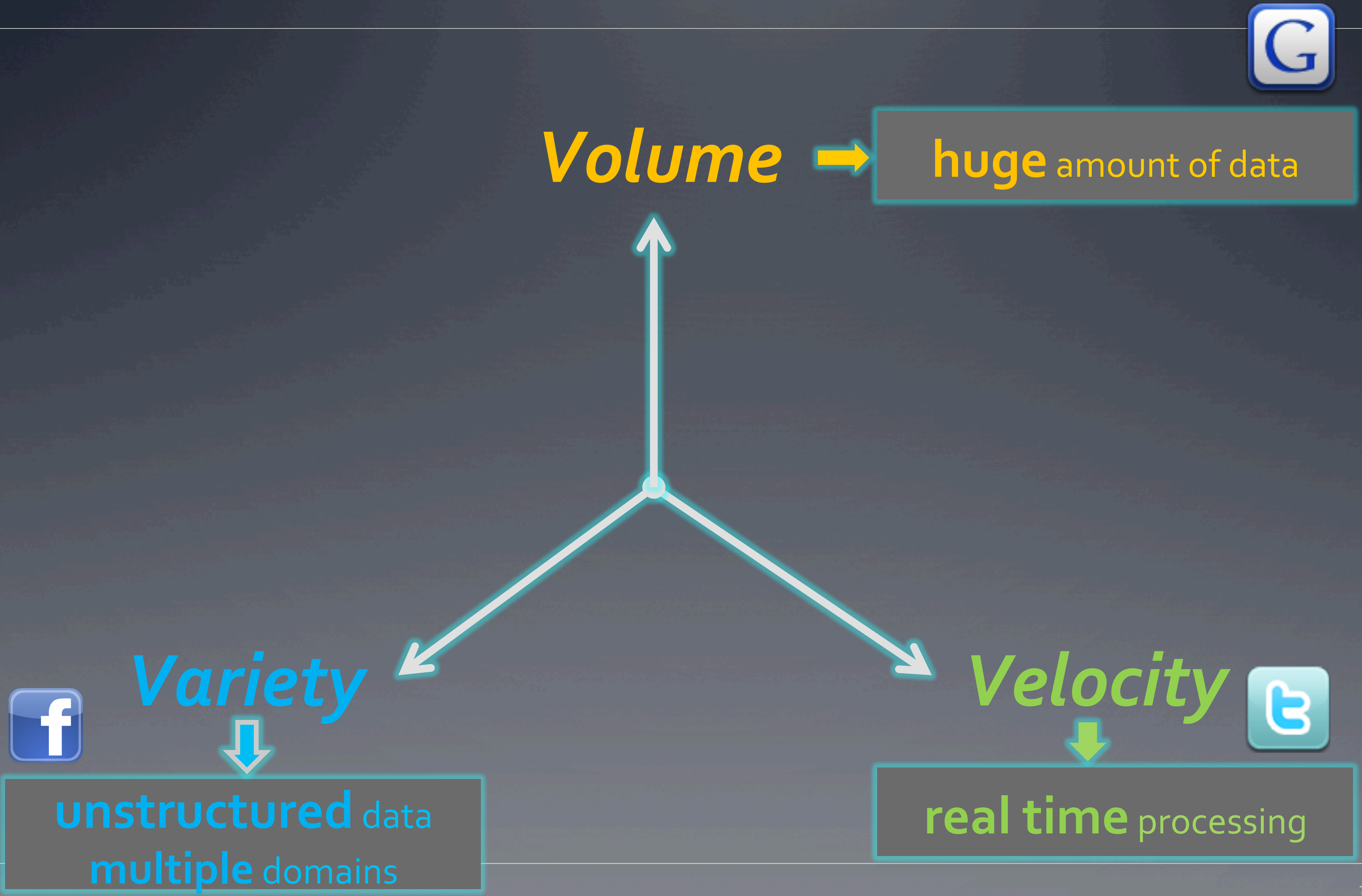
Review: Primary trends in forensics



looks like we have a “big data” problem!

The three "V"s of big data

(by *Michael Stonebraker*)

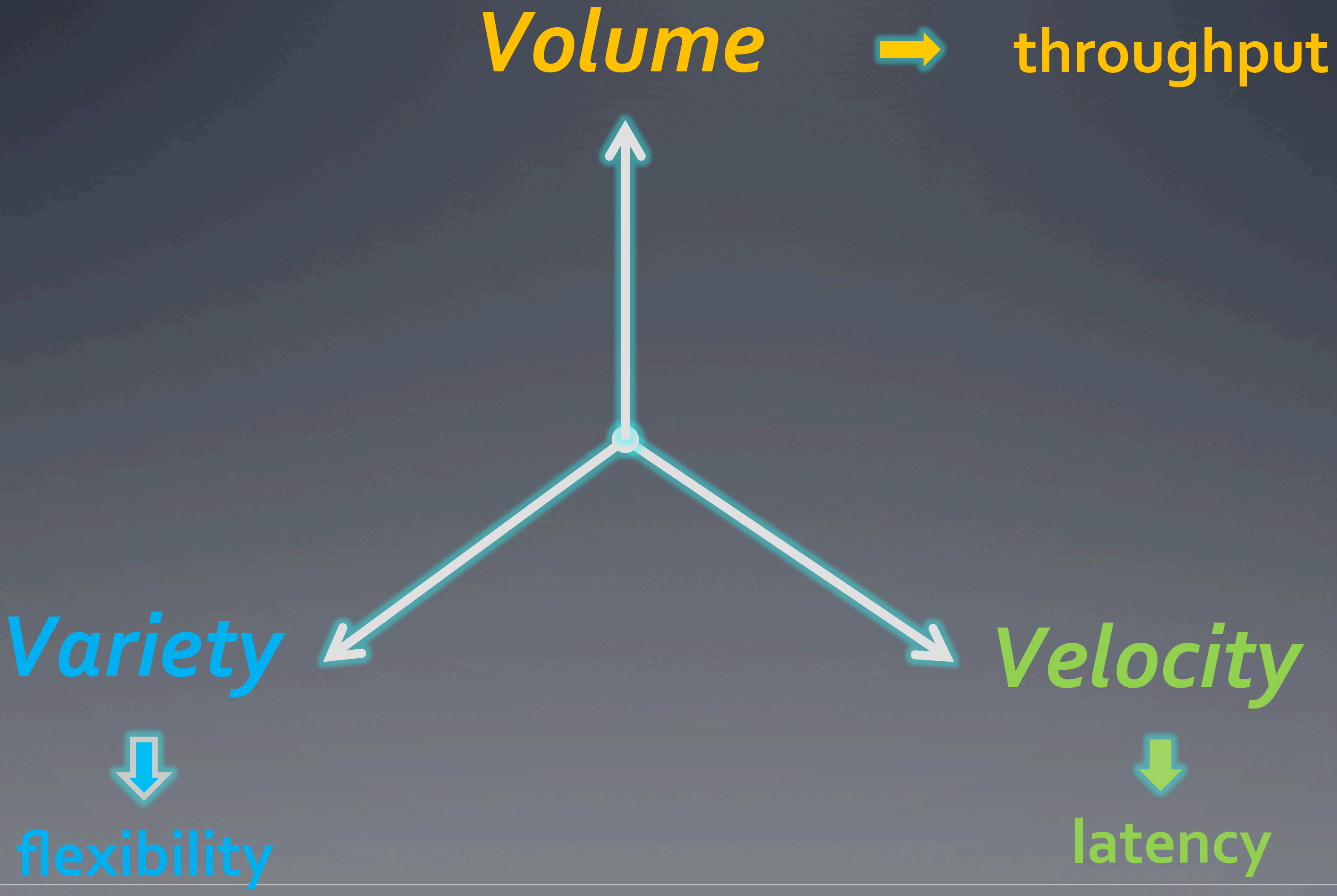


Volume → **huge** amount of data

Variety ⇩ **unstructured** data **multiple** domains

Velocity ⇩ **real time** processing

Optimization priorities

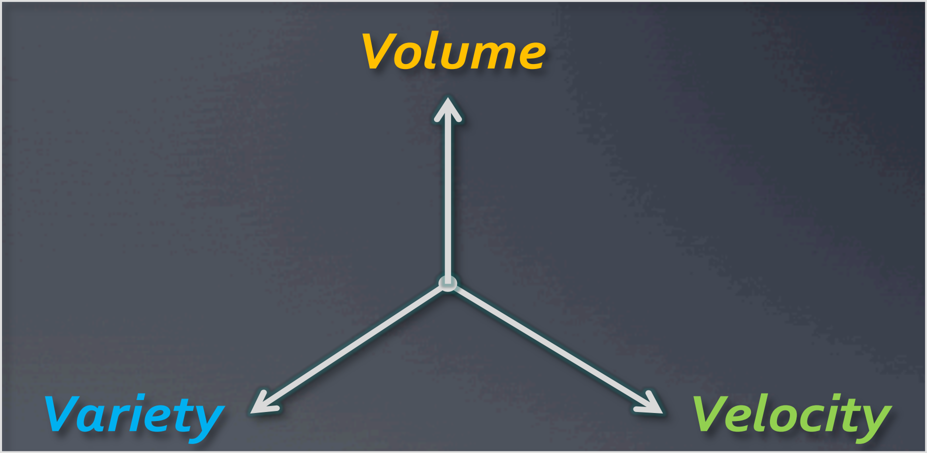


Volume ➡ throughput

Variety ⇩ flexibility

Velocity ⬇ latency

What is forensics' *primary* challenge?



- Volume:
  - Do we have PB of data?

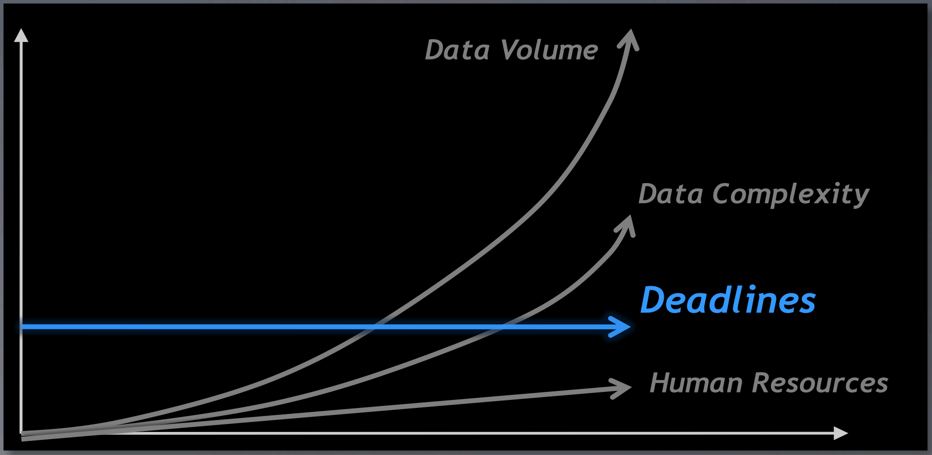
Not really!

- Variety:
  - How many *types* of things do we need to process?

Few (but growing)

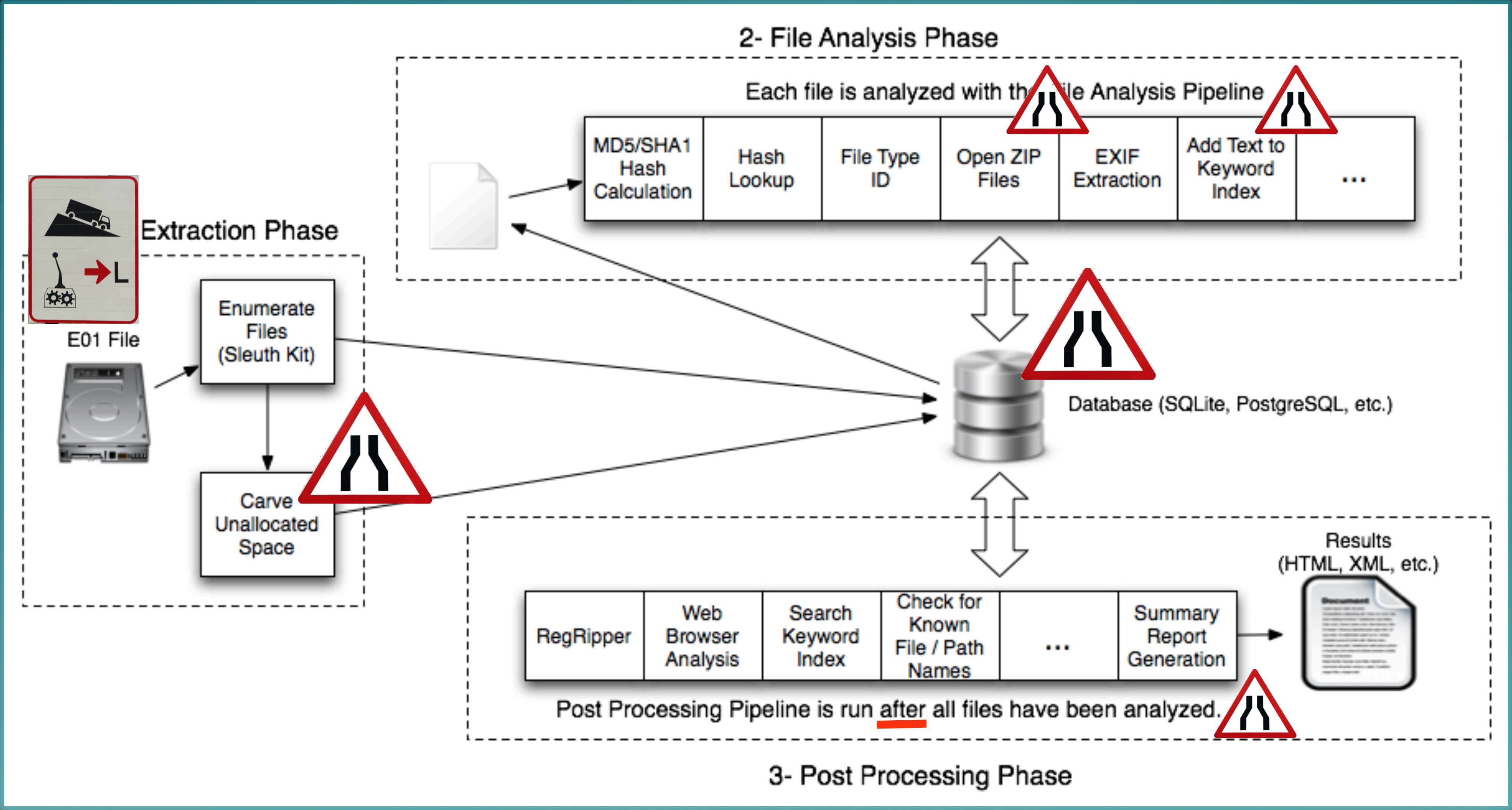
- Velocity:
  - Do we have deadlines?

YES!!!



Apple/Jeremy Johnstone/Twitter

Where are we now?



2- File Analysis Phase

Each file is analyzed with th le Analysis Pipeline

MD5/SHA1 Hash Calculation

Hash Lookup

File Type ID

Open ZIP Files

EXIF Extraction

Add Text to Keyword Index

...

Extraction Phase

E01 File

Enumerate Files (Sleuth Kit)

Carve Unallocated Space

Database (SQLite, PostgreSQL, etc.)

Results (HTML, XML, etc.)

RegRipper

Web Browser Analysis

Search Keyword Index

Check for Known File / Path Names

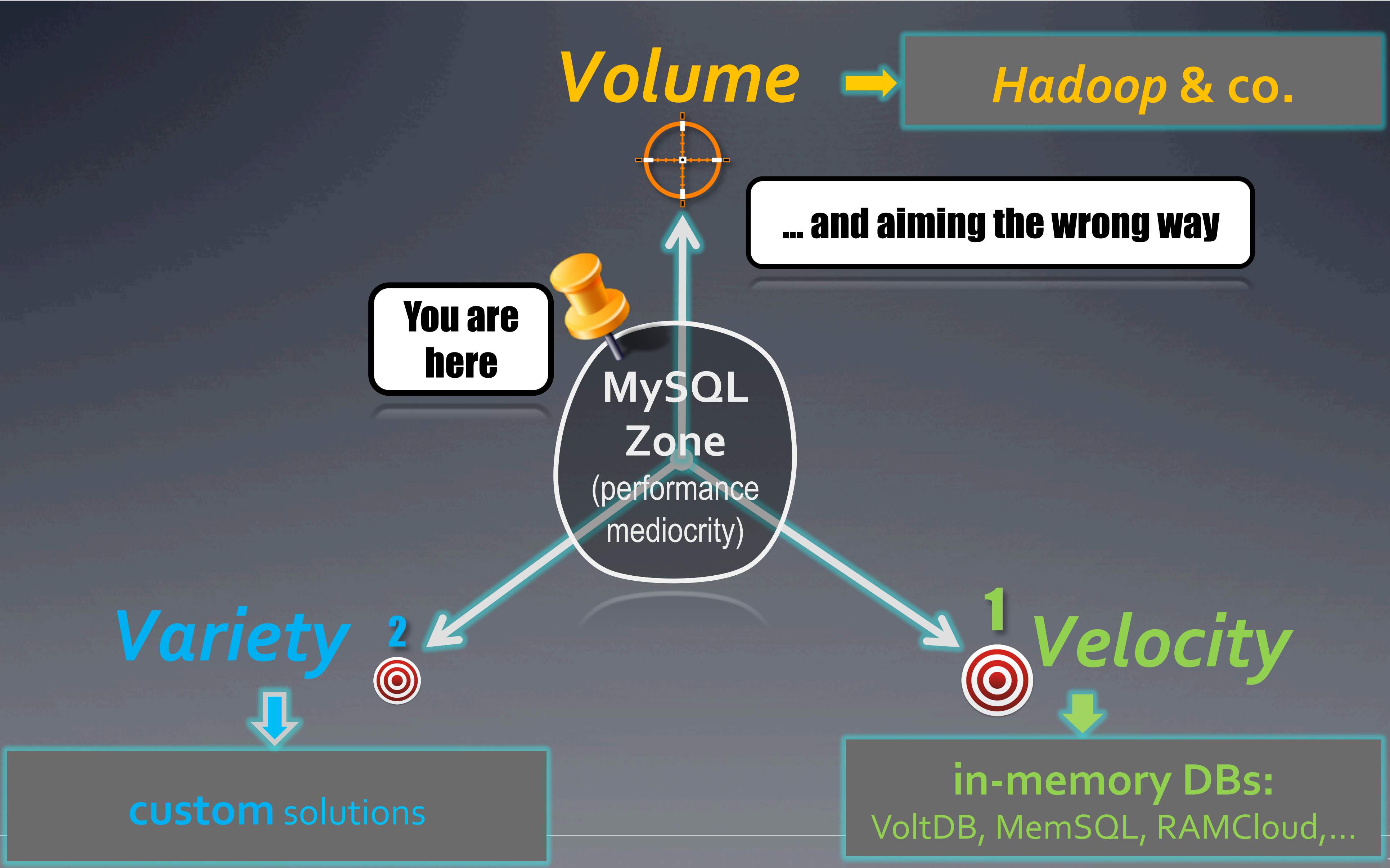
...

Summary Report Generation

Post Processing Pipeline is run after all files have been analyzed.

3- Post Processing Phase

Mapping problems to solutions



Volume → *Hadoop & co.*

… and aiming the wrong way

You are here

MySQL Zone (performance mediocrity)

Variety 2

custom solutions

1 *Velocity*

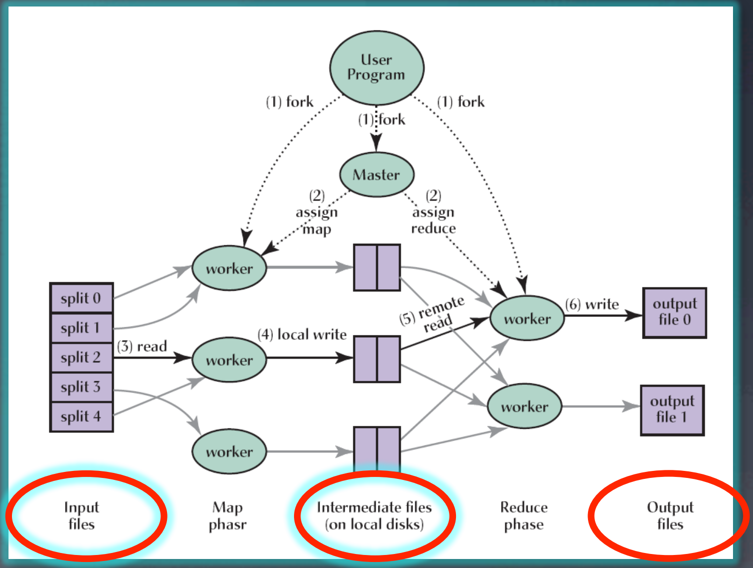
in-memory DBs:
VoltDB, MemSQL, RAMCloud,…

What's wrong with Hadoop?

- Nothing

  … if you have a LOT of data (100TB^{+})

  … it's your only choice, really



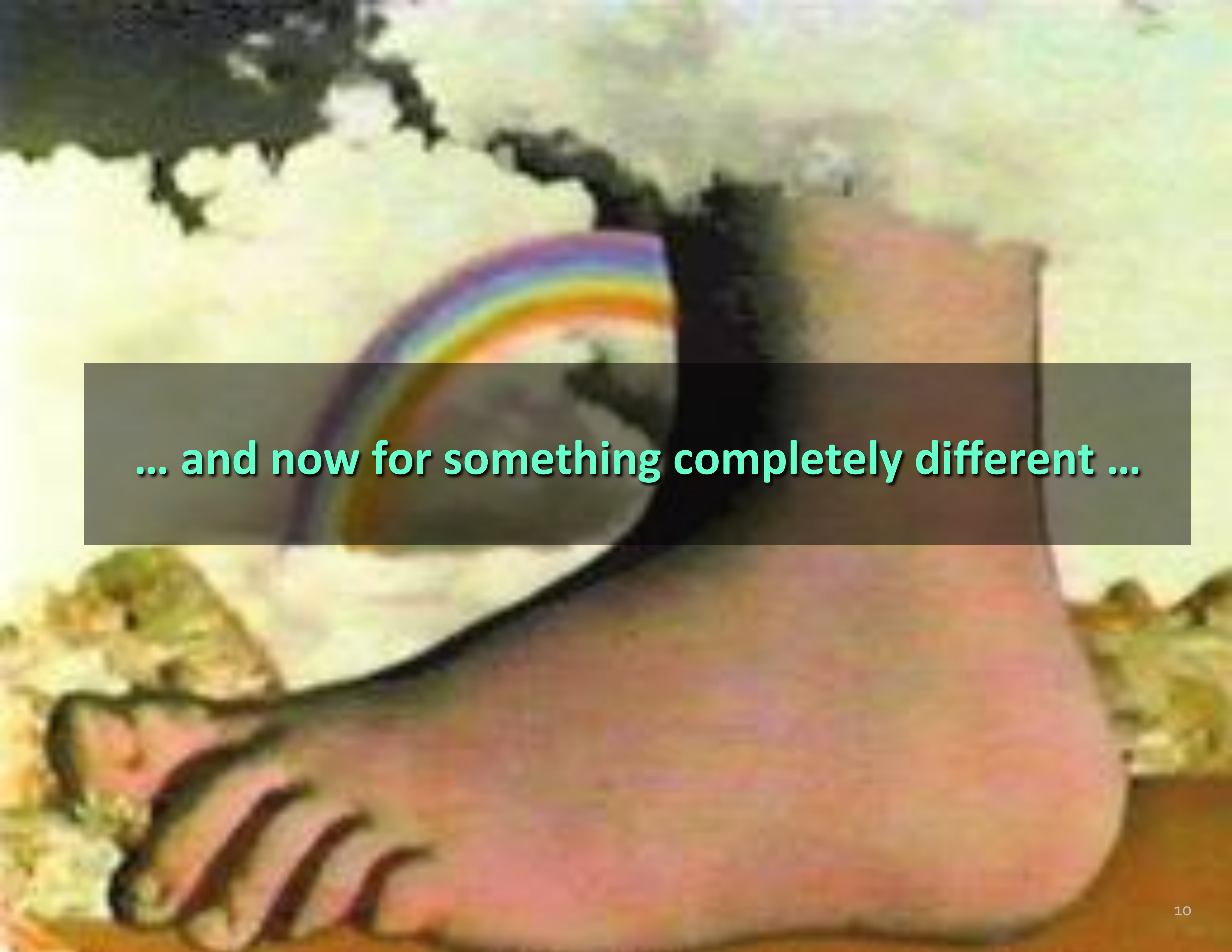
- If you don't?

  … you still have to wait

- It is a throughput engine
  - Requires a lot of time to seed initially (HDFS)
  - Suitable for data processing sweeps over *entire* sets
  - Tasks communicate via the file system
  - Not all processing fits the M/R model
  - Will do ***nothing*** to speed up triage and early processing

In other words …

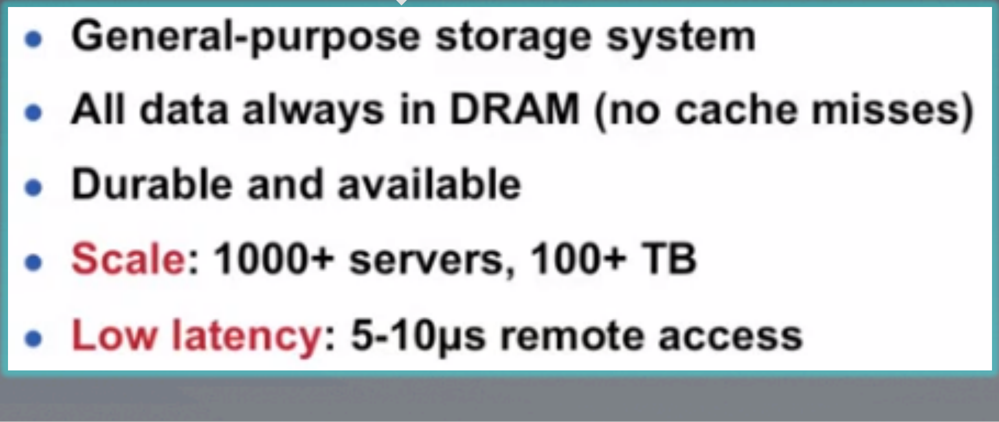
- The failure of current tools to address latency requirements leads to data backlogs.
- This leads to the ***perception*** that we have a volume problem.
- Using a “bigger hammer” designed for volume will do little to address latency.



… and now for something completely different …

Elsewhere …
“big data” world is moving into RAM

- 2003: All Web indexes are served from RAM
- 2009: At Facebook 150 out of 200 TB cached
- New RAM data stores (***not*** caches)
  - Commercial: MemSQL, VoltDB, SQLFire,
  - Research: RAMCloud, H-Store, HyperDex



- General-purpose storage system
- All data always in DRAM (no cache misses)
- Durable and available
- Scale: 1000+ servers, 100+ TB
- Low latency: 5-10µs remote access

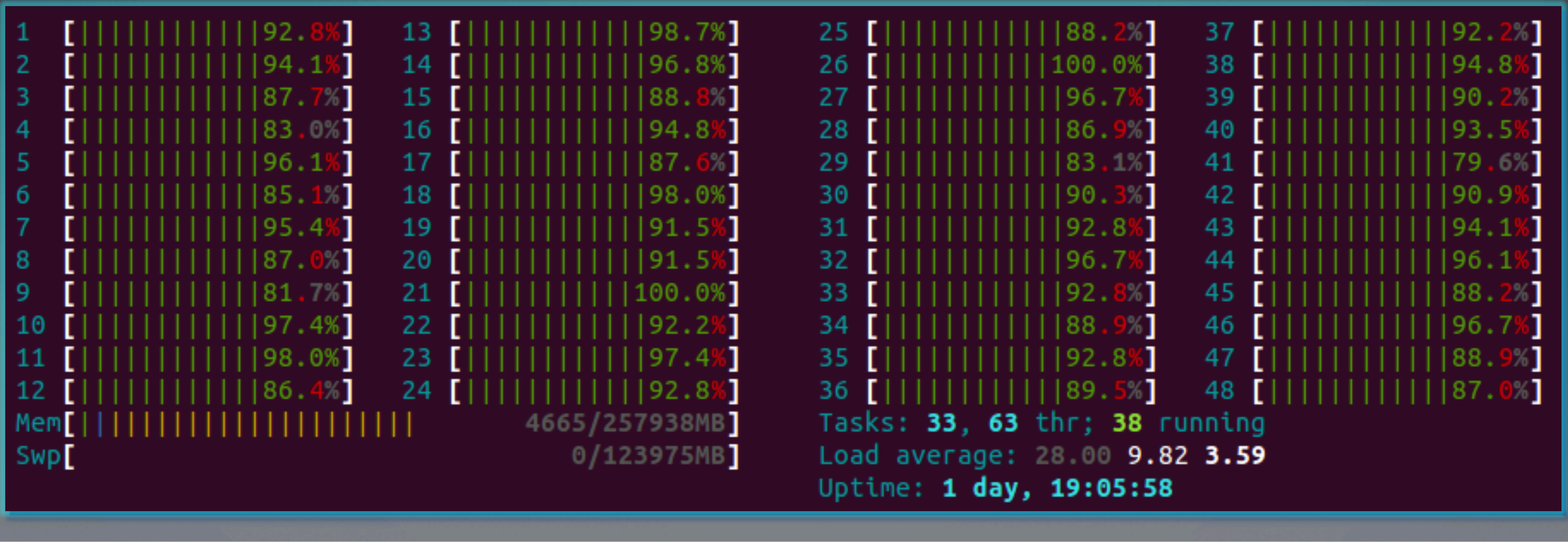
“Say ‘hello’ to my little friend”

- Dell PowerEdge R815
  - 48 cores @2.6GHz AMD
  - 256 GB RAM
  - 10Gb Ethernet
- Price?
  - 13-18 *iPads* !!

- 4 x R815 == neat little cluster:
  - 192 cores
  - **1TB** RAM

Fun things to do on 48 cores
(and 256GB of RAM)

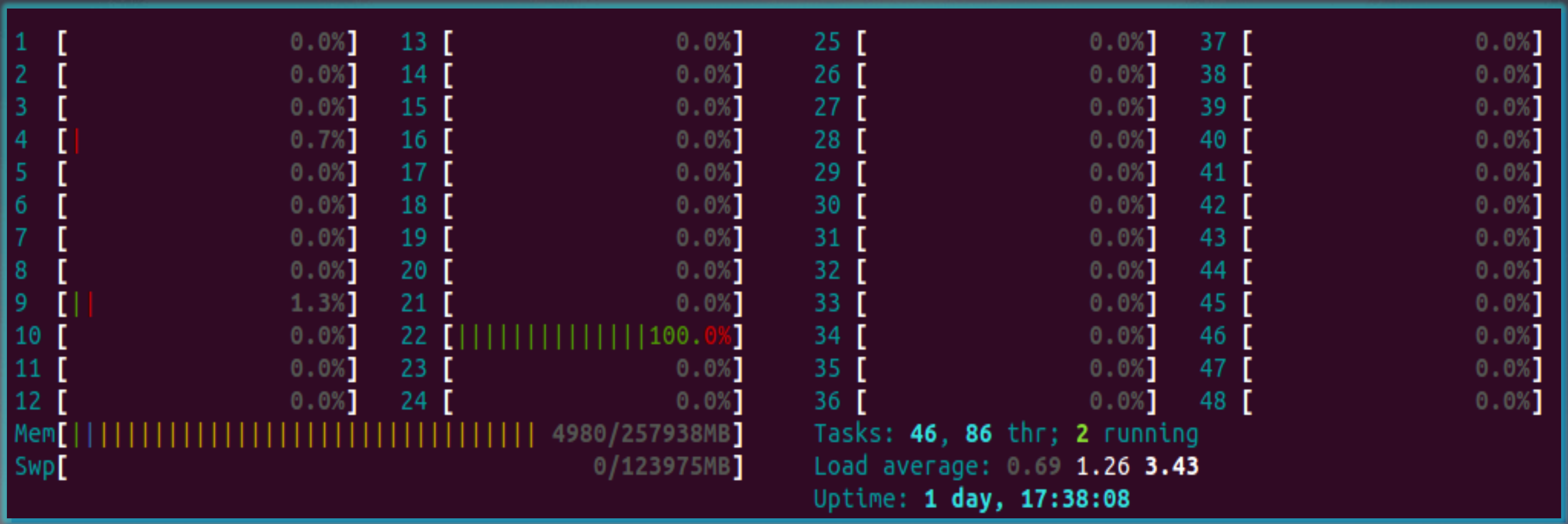
- `pbzip2 -p48 target.dd` → 272MB/s
- `pbzip2 -d -p48 target.dd.bz2` → 677MB/s
- `pigz -p 48 target.dd` → 832MB/s



```
1   [||||||||||||92.8%]   13 [||||||||||||98.7%]     25 [||||||||||||88.2%]   37 [||||||||||||92.2%]
2   [||||||||||||94.1%]   14 [||||||||||||96.8%]     26 [|||||||||||100.0%]   38 [||||||||||||94.8%]
3   [||||||||||||87.7%]   15 [||||||||||||88.8%]     27 [||||||||||||96.7%]   39 [||||||||||||90.2%]
4   [||||||||||||83.0%]   16 [||||||||||||94.8%]     28 [||||||||||||86.9%]   40 [||||||||||||93.5%]
5   [||||||||||||96.1%]   17 [||||||||||||87.6%]     29 [||||||||||||83.1%]   41 [||||||||||||79.6%]
6   [||||||||||||85.1%]   18 [||||||||||||98.0%]     30 [||||||||||||90.3%]   42 [||||||||||||90.9%]
7   [||||||||||||95.4%]   19 [||||||||||||91.5%]     31 [||||||||||||92.8%]   43 [||||||||||||94.1%]
8   [||||||||||||87.0%]   20 [||||||||||||91.5%]     32 [||||||||||||96.7%]   44 [||||||||||||96.1%]
9   [||||||||||||81.7%]   21 [|||||||||||100.0%]     33 [||||||||||||92.8%]   45 [||||||||||||88.2%]
10  [||||||||||||97.4%]   22 [||||||||||||92.2%]     34 [||||||||||||88.9%]   46 [||||||||||||96.7%]
11  [||||||||||||98.0%]   23 [||||||||||||97.4%]     35 [||||||||||||92.8%]   47 [||||||||||||88.9%]
12  [||||||||||||86.4%]   24 [||||||||||||92.8%]     36 [||||||||||||89.5%]   48 [||||||||||||87.0%]
Mem[||||||||||||||||||||||        4665/257938MB]     Tasks: 33, 63 thr; 38 running
Swp[                                 0/123975MB]     Load average: 28.00 9.82 3.59
                                                     Uptime: 1 day, 19:05:58
```

Unfun things to do on 48 cores
(and 256GB of RAM)

- ewfacquire … target.dd → 74MB/s
- ewfexport … target.E01 → 147 MB/s

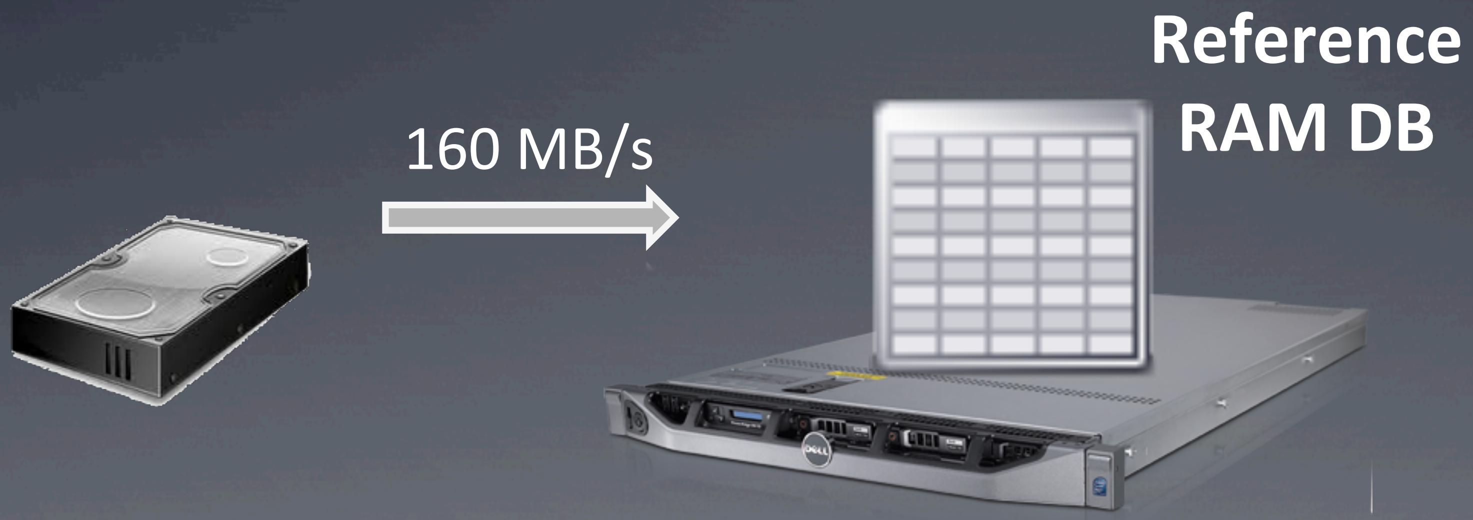


```
1  [                 0.0%]  13 [                 0.0%]    25 [                 0.0%]   37 [                 0.0%]
2  [                 0.0%]  14 [                 0.0%]    26 [                 0.0%]   38 [                 0.0%]
3  [                 0.0%]  15 [                 0.0%]    27 [                 0.0%]   39 [                 0.0%]
4  [|                0.7%]  16 [                 0.0%]    28 [                 0.0%]   40 [                 0.0%]
5  [                 0.0%]  17 [                 0.0%]    29 [                 0.0%]   41 [                 0.0%]
6  [                 0.0%]  18 [                 0.0%]    30 [                 0.0%]   42 [                 0.0%]
7  [                 0.0%]  19 [                 0.0%]    31 [                 0.0%]   43 [                 0.0%]
8  [                 0.0%]  20 [                 0.0%]    32 [                 0.0%]   44 [                 0.0%]
9  [||               1.3%]  21 [                 0.0%]    33 [                 0.0%]   45 [                 0.0%]
10 [                 0.0%]  22 [||||||||||||||100.0%]     34 [                 0.0%]   46 [                 0.0%]
11 [                 0.0%]  23 [                 0.0%]    35 [                 0.0%]   47 [                 0.0%]
12 [                 0.0%]  24 [                 0.0%]    36 [                 0.0%]   48 [                 0.0%]
Mem[|||||||||||||||||||||||||||||||||| 4980/257938MB]     Tasks: 46, 86 thr; 2 running
Swp[                                      0/123975MB]     Load average: 0.69 1.26 3.43
                                                          Uptime: 1 day, 17:38:08
```

Useful things to do with 48 cores

(and 256GB of RAM)

- Screen content of a target *at line speed* with similarity digests (*sdhash 3.0alpha*):



In V3.0 (Oct ‘12), we will cover up to 10GB of source data (RefDB: ~500MB)

In V3.1 (Dec’12), source data should be in the 1 to 10 TB range (RefDB: ~50-500GB)

Useful things to do with 48 cores

(and 256GB of RAM)

- Index (w/ Solr) 100GB in 40min
  - 43MB/s
  - Tested on GovDocs files: txt/html/doc/pdf/ppt/…
  - Zero disk I/O

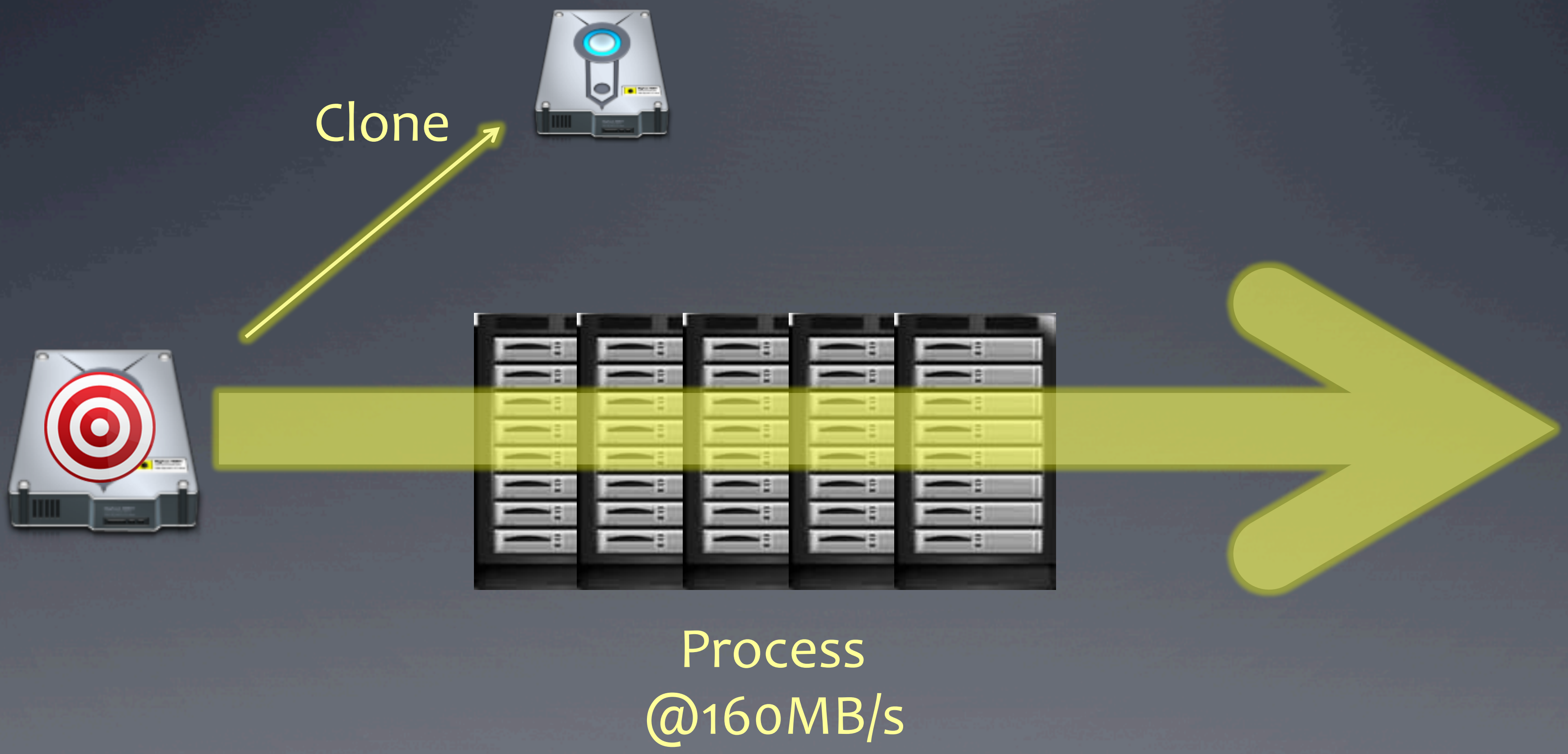
```
time parallel exiftool -- /corpora/nps-gov/00?/*
real    0m57.368s
user   37m41.317s
sys     2m52.163s
```

10,000 files, 5.5GB, cached

```
time parallel exiftool -- /corpora/nps-gov/01?/*
real    1m37.142s
user   30m51.704s
sys     2m46.230s
```

10,000 files, 5.0GB, on disk

End goal: Real-time forensic processing



Objective: Finish cloning & processing at the same time.

Real-time forensics "showstoppers"

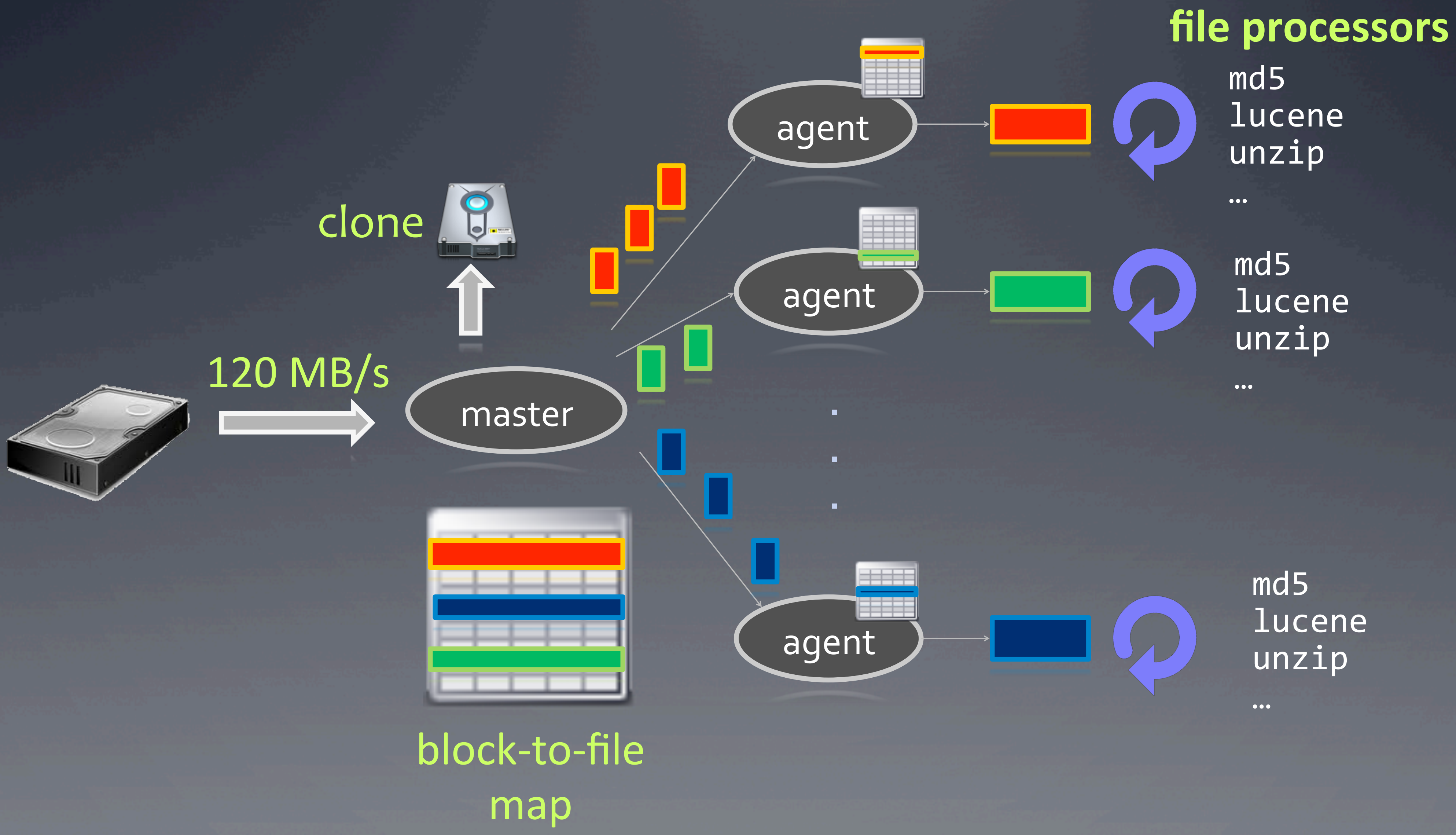
- **File**-based processing
  - hashing, metadata extraction, thumbnailing, …
  - generates non-sequential access & horrible I/O
- Indexing
  - search engines optimize query performance, not indexing
- Carving
  - can generate ***huge*** amounts of false positives (& potentially nasty I/O)

Latency-optimized target acquisition

- The problem
  - Most processing is file-centric
  - File-based access —> bad I/O on HDD
- Solution sketch
  - Before imaging, map blocks to files (45sec for 186,000 files)
  - During imaging, incrementally reconstruct files
    - Using multiple (potentially distributed) agents
  - Once file is complete, make it available via file system
  - File-based tool can pick it up and process it as usual
- End game
  - Given enough RAM/CPUs, time(cloning) == time(processing)

LOTA implementation

(by *Rob Martell*)



file processors

md5
lucene
unzip
…

md5
lucene
unzip
…

md5
lucene
unzip
…

agent

agent

agent

clone

120 MB/s

master

block-to-file
map

The takeaway (1)

- The primary performance concern of DF is ***latency***
  - Volume accumulation is a symptom of bad tools
  - Forensics needs to move to a real time model

    ➔ **real-world processing has deadlines**

- Forensic analysis must start at conception
  - Move from 'clone-first' to 'latency-first' processing

- RAM will save the day (not *Hadoop*)
  - 10-1000x speedup for I/O-bound processing
  - Enables massive parallel processing

The takeaway (2)

- Current uses for (clusters of) high-RAM boxes
  - sdhash-based screening (NSRL scale)
  - latency optimized target acquisition (LOTA?)
  - indexing
  - metadata extraction, zip/unzip, thumbnailing
  - bulk_extractor
  - MySQL —> MemSQL
- We need a new platform to make all of this easy & extensible
  - We're working on it …

Thank You!

- Q & A
- Contact

  Vassil Roussev

  `vassil@roussev.net`
- sdhash 3.0 (exp. 10/15)

  `sdhash.org`