Software Execution Protection in the Cloud

Miguel Correia

1st European Workshop on Dependable Cloud Computing
Sibiu, Romania, May 8th 2012

INSTITUTO SUPERIOR TÉCNICO

FCT Fundação para a Ciência e a Tecnologia

inesc id lisboa

Motivation – clouds fail

Motivation – accidental arbitrary faults

- Recent studies:
  - Google datacenters: more than 8% DIMMs affected by errors yearly (even with ECC)
  - Consumer PCs (Microsoft): CPU and core chipset faults are frequent
- Known cases:
  - Sun server caches corrupted by cosmic rays (~2000)
  - IOMMU in AMD chipsets corrupted data due to a bug activated by certain sw/hw combinations
- These faults can corrupt software execution

3

Motivation – malicious faults

- Recent studies:
  - Malicious insiders with access to servers' management VM (dom 0 in Xen) can easily obtain passwords, private RSA keys, files, thus tamper with user VMs
- Recent cases:
  - Google engineer read Gmail/Gtalk communications and contacted teen users
  - CyberLynk (cloud storage) ex-employee deleted a season of a kids TV series
- These attacks can corrupt software execution

4

Outline

- Protecting MapReduce executions from accidental arbitrary faults
- Protecting user virtual machines from malicious faults
- Conclusions

5

PROTECTING MAPREDUCE EXECUTIONS FROM ACCIDENTAL ARBITRARY FAULTS

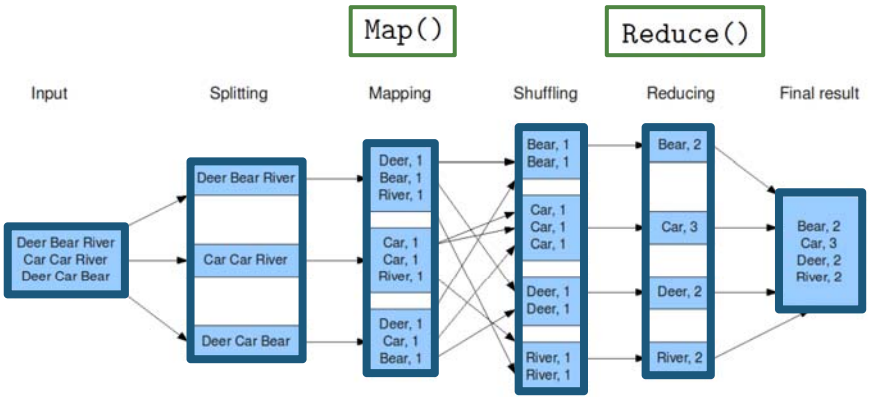
6

What is MapReduce?

- Programming model + execution environment
  - Introduced by Google in 2004
  - Used for processing large data sets using clusters of servers
  - Currently several implementations are available and it's used by many organizations
- Hadoop MapReduce, an open-source MapReduce
  - Probably the most used, the one we used
  - Includes the Hadoop Distributed File System (HDFS) – a file system for GB files

7

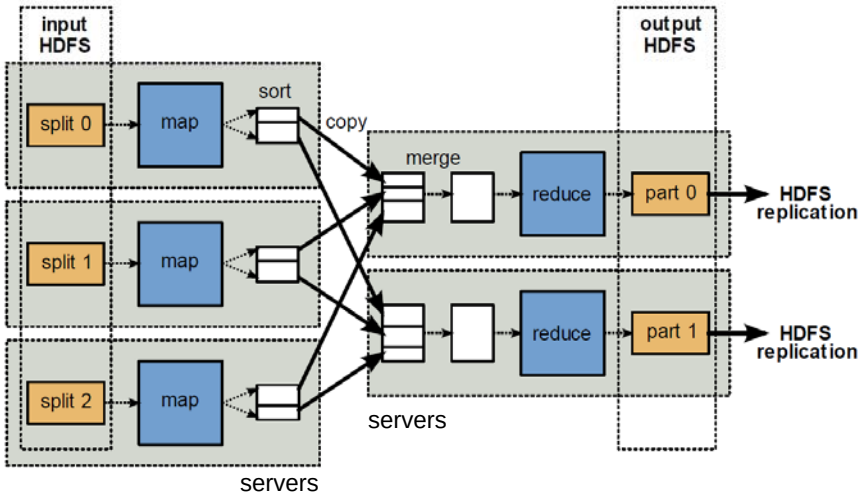
WordCount example



GB files!

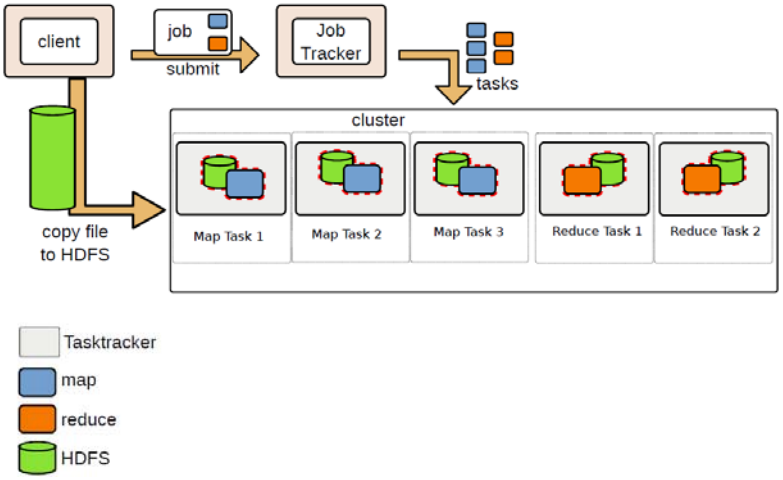
8

Data flow and locality



9

Job submission and execution



10

Fault Tolerance

- The original Hadoop MR tolerates some faults
  - Job tracker detects and recovers crashed map/reduce tasks
  - Detects corrupted files (a hash is stored with each block)
- But execution can be corrupted and a task can return wrong output
  - e.g., due to memory corruption or chipset errors

11

Accidental arbitrary fault tolerance

- (or Byzantine fault tolerance – BFT)
- Basic approach is to replicate task execution and vote
  - Given *f* = the maximum number of faulty replicas
- Standard approaches:
  - Replication + consensus → all tasks executed *3f+1* times
  - Replication + client voting → all tasks executed *2f+1* times
  - Computation multiplied by at least 3!
- Our approach:
  - All tasks executed *f+1* times *+ 1* task execution/fault
  - Tolerates more than *f* faults (meaning of *f* is slightly different)

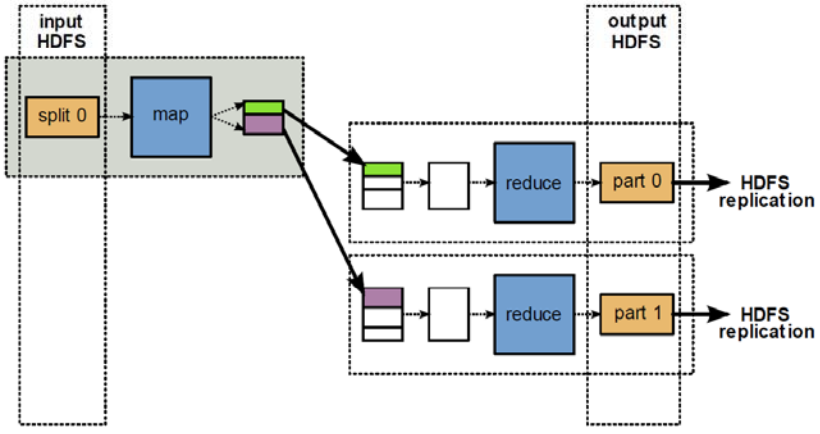
12

System model

- Tasks (map/reduce) and nodes can be correct or faulty
- Client is correct (not part of MR)
- Job Tracker is correct (like in Hadoop MR)
- Tasks: for every task, at most *f* of its *faulty replicas* can produce the same output
  - Otherwise the number of faulty replicas is unbounded
- Next: basic idea

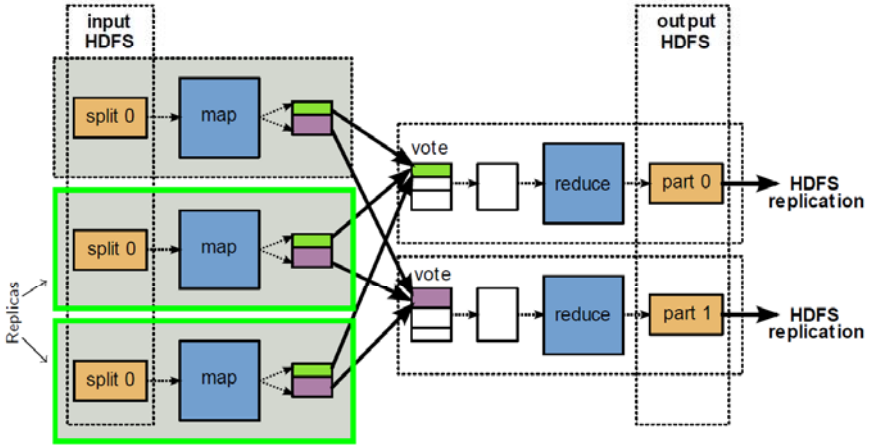
13

Original MR – Map perspective



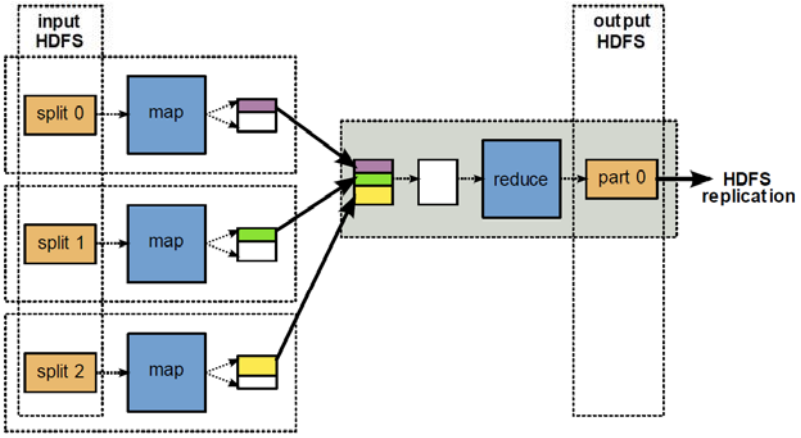
14

Basic BFT MR – Map perspective



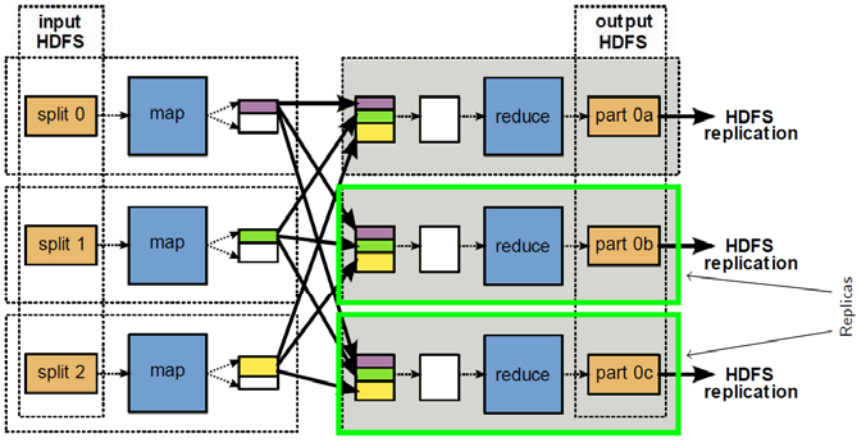
15

Original MR – Reduce perspective



16

Basic BFT MR – Reduce perspective



17

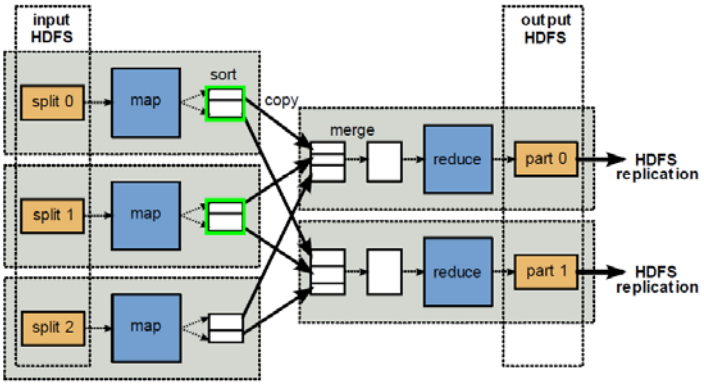
Improvements over basic version

- Basic version: map/reduce tasks executed *2f+1* times
- Our BFT MR can be thought of as this basic version plus the following modifications:
  - Deferred execution
  - Tentative reduce execution
  - Digest outputs
  - Tight storage replication

18

Deferred execution

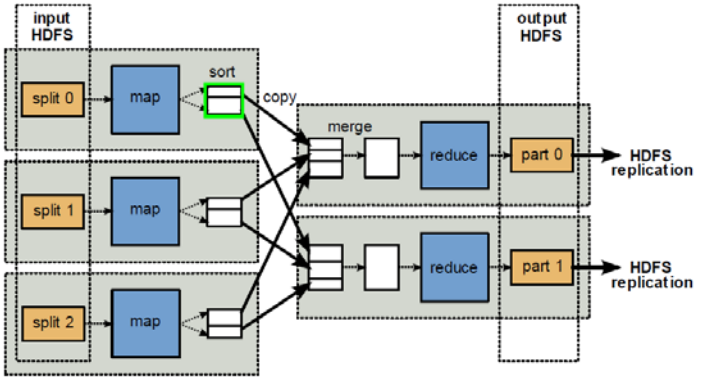
- Faults are uncommon
- Job Tracker creates only *f+1* replicas
- Creates more if the results aren't equal (until *f+1* are equal)



19

Tentative reduce execution

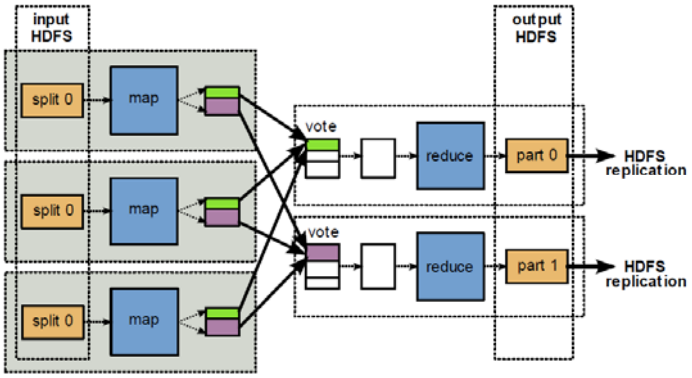
- Start reduce tasks when the first input becomes available (instead of waiting for *f+1* equal inputs)
- Restart reduce tasks if inputs disagree



20

Digest outputs

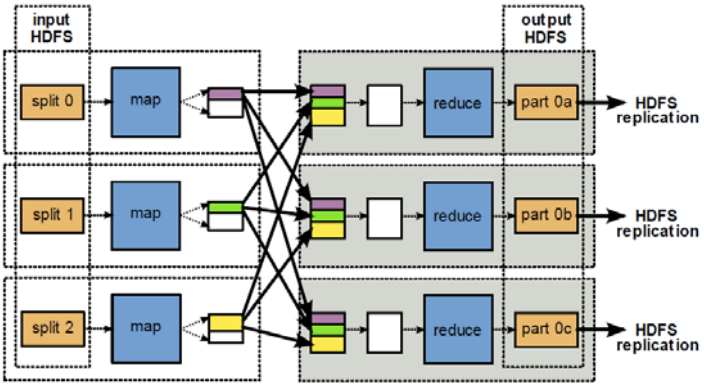
- Fetch data only from one map replica
- Fetch a digest (hash) from the remaining ones



21

Tight storage replication

- HDFS replicates data blocks for fault tolerance
- Turn off HDFS replication for Reduce output data



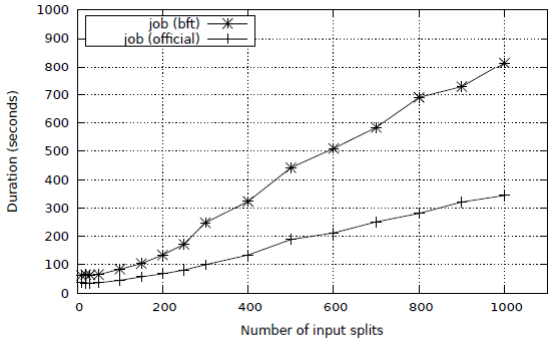
22

Experimental evaluation

- GridMix benchmark
  - Standard benchmark for Hadoop MapReduce
  - 6 different types of applications
- Executed in the Grid'5000 infrastructure (France)
  - No faults injected
  - *f=1* – twice as much computation as the original MR
  - 10 and 20 nodes of the same type
  - Variable number of input splits (hundreds of MB to GB)
  - Ran 5 times each case; results are average

23

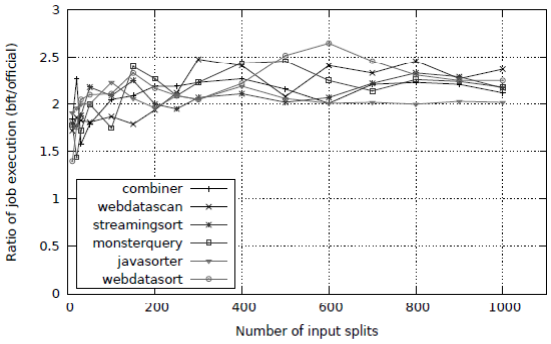
Job Execution Time – 10 nodes



- WebdataScan app. with variable number of 64 MB input splits
- BFT version took around the double of time

24

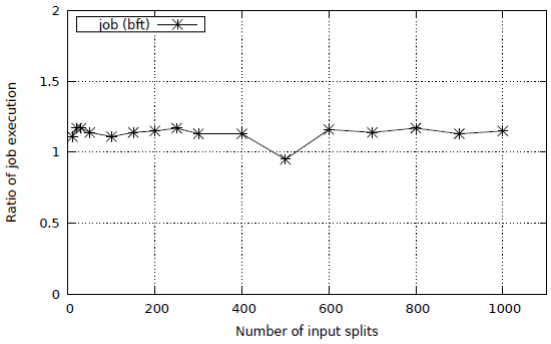
BFT/original exec. times – 10 nodes



- Ratio of duration of BFT and original versions
- BFT version takes from 1.5 to 2.5 more time

25

BFT/original exec. times – 20 nodes



- With more nodes, BFT takes only slightly longer than original
- But CPU time used is still the double

26

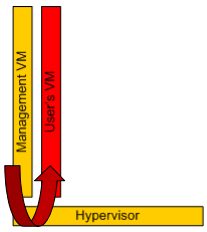
PROTECTING USER VIRTUAL MACHINES FROM MALICIOUS FAULTS

27

Malicious insider example attack

- Consider an IaaS cloud that runs user VMs
  - Servers contain hypervisor, mgmt VM, and run user VMs
- Malicious insider can attack VMs
  - by abusing server functionality, e.g., memory snapshot
- Attack – insider logs in Xen dom 0 and runs:

```
$ xm dump-core 2 -L lucidomu.dump
Dumping core of domain: 2 ...
$ rsakeyfind lucidomu.dump
found private key at 1b061de8
…
```



28

Protecting user VMs – basic idea

- To prove to the cloud user that its data is in a server with a safe software configuration
  - e.g., in which the management VM has no snapshot function
- Do this using the Trusted Platform Module (TPM), a security chip from the Trusted Computing Group
  - now shipping with common PC hardware

29

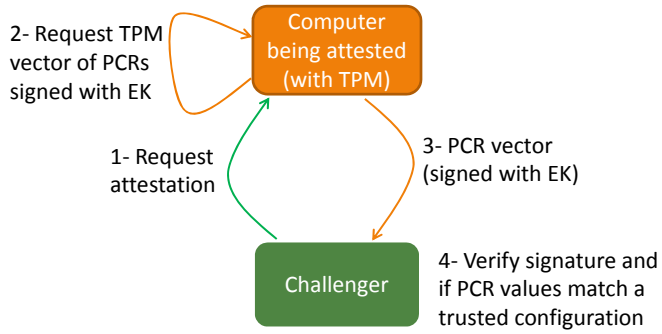
Measurements

- TPM has Platform Configuration Registers (PCR)
- A PCR stores (typically) a measurement of a software block, i.e., its cryptographic hash
  - During system boot, BIOS stores *hash(boot loader)* in PCR_0, boot loader stores *hash(hypervisor)* in PCR_1, …
- A vector of PCR values gives a trusted measurement of the software configuration
  - It is signed by the TPM's Endorsement Key (EK)
  - EK certificate signed by TPM's vendor (means it's a real TPM)

30

Remote attestation

- Computer gives to a challenger a measurement of the software configuration (vector of PCR values)



31

Approach overview

- Servers run a Trusted Virtualization Environment (TVE), formed by hypervisor + management VM that the user trusts
  - TVE does not provide dangerous operations to administrators: memory snapshot, volume mount
  - TVE provides only trusted versions of certain operations
  - VMs enter and leave a TVE encrypted
- Users do remote attestation of TVEs/operations to be sure that their VMs are either in a TVE or encrypted
  - The environment is a TVE if its measurements (PCR values) fall in a set of TVE-configurations

32

Limitations of TVEs

- TVEs apparently solve the problem…
  - User VMs run only on TVEs so they are secure
- …but
  - A TVE is too big (is it possible to trust ~400 KLOCs?)
  - TPM is too slow to run attestations with several VMs per server (e.g., can do only ~2 per second!)

33

Trusted Cloud Operation Modules

- TCOMs are modules that implement trusted versions of certain operations: launch, migrate,…
  - They're specific modules thus much smaller than a TVE
- TCOMs are executed in an Isolated Execution Environment (IEE)
  - An IEE is created in runtime based on a DRTM (TrustVisor)
- TCOM measurements are extended into a µTPM
  - µTPM is software, but much faster than the TPM
- Smaller, faster: solves the 2 limitations of TVEs

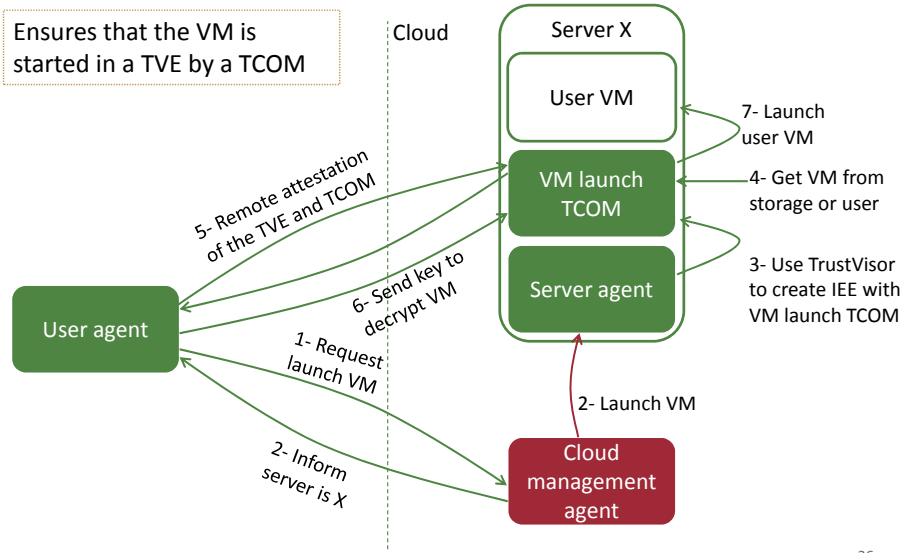
34

Trusted VM operations

- Operations that have to be trusted (so need TCOMs):
  - VM launching, VM migration, VM backup, VM termination
- These operations involve four entities
  - Server agent – trusted because it is in the TVE
  - TCOM of the operation in a server – trusted
  - User agent – trusted, not part of the cloud
  - Cloud management agent – not trusted, what the malicious insider may control

35

VM launching



36

VM migration, backup, terminate

- The process is similar for all except:
- VM migration
  - The TCOM of the destination server has also to be attested
  - Source TCOM encrypts VM with session key; destination TCOM decrypts it
- VM backup
  - TCOM encrypts the VM
  - TCOM gives the key to the user agent
- VM termination
  - TCOM scrubs the memory and disk to delete all VM data

37

Open problems

- Gap between checking a measurement (just a hash) and trusting a complex software module (TVE)
  - How can we know that there isn't really undesirable functionality, vulnerabilities, or malware inside?
- Putting this solution in production
  - Short time to market and many players: cloud provider, software producers, assurance labs, cloud user

38

CONCLUSIONS

39

Conclusions

- Software executed in the cloud can be corrupted by:
- arbitrary accidental faults
  - Memory corruptions
  - Other hardware faults
- malicious faults / attacks
  - Malicious insiders
  - And many others…

40

Conclusions

- Tolerance to accidental faults in MapReduce
  - Execute each task more than once and compare results
  - Do it efficiently to multiply computation by only 2, with *f=1* (with no faults)
- Protection from malicious insiders
  - Use trusted computing / the TPM to create root of trust
  - Cloud providers may implement something of the kind soon (TCG, Intel, IBM are pushing)

41

Questions?