University of Nevada, Reno

# Is End-to-End Integrity Verification Really End-to-End?

Ahmed Alhussen, Batyr Charyyev, and **Engin Arslan**

# What’s End-to-End Integrity Verification

- **Data corruption may occur during transfers**
  - Faulty equipment, transient errors etc.
- **Existing integrity check mechanisms are weak**
  - TCP checksum fails to once in every 16 million to 10 billion packets
- **End-to-end integrity verification offers strong fault-tolerance guarantee**
  - Secure hash algorithms SHA1, SHA-254
  - Captures errors that could happen anywhere during transfers; network, server, and disk (?)

# How End-to-End Integrity Verification Works?

| Sender | Receiver |
|---|---|
| Read the file and send | Receive the file and save |

# How End-to-End Integrity Verification Works?

| Sender | Receiver |
|---|---|
| Read the file and send | Receive the file and save |
| Read the file again and compute checksum | Read file back from storage and compute checksum |

# How End-to-End Integrity Verification Works?

| Sender | Receiver |
|---|---|
| Read the file and send | Receive the file and save |
| Read the file again and compute checksum | Read file back from storage and compute checksum |
| Accept receiver’s checksum | Send checksum |

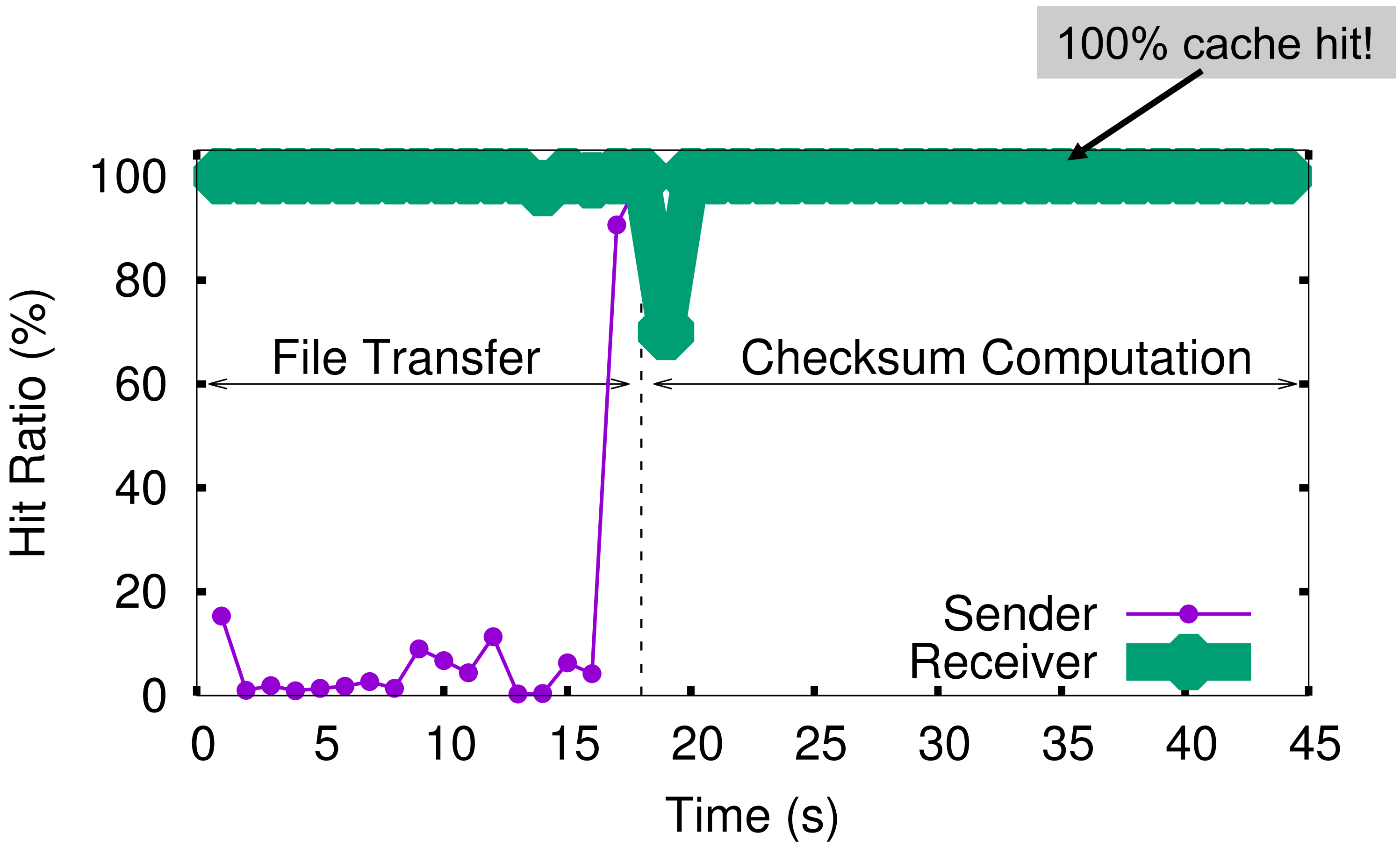
# How End-to-End Integrity Verification Works?

| Sender | Receiver |
|---|---|
| Read the file and send | Receive the file and save |
| Read the file again and compute checksum | Read file back from storage and compute checksum |
| Accept receiver’s checksum | Send checksum |
| If checksums match→ Done | |

# How End-to-End Integrity Verification Works?

| Sender | Receiver |
|---|---|
| Read the file and send | Receive the file and save |
| Read the file again and compute checksum | Read file back from storage and compute checksum |
| Accept receiver’s checksum | Send checksum |
| If checksums match→ Done | |
| Else →Transfer again | |

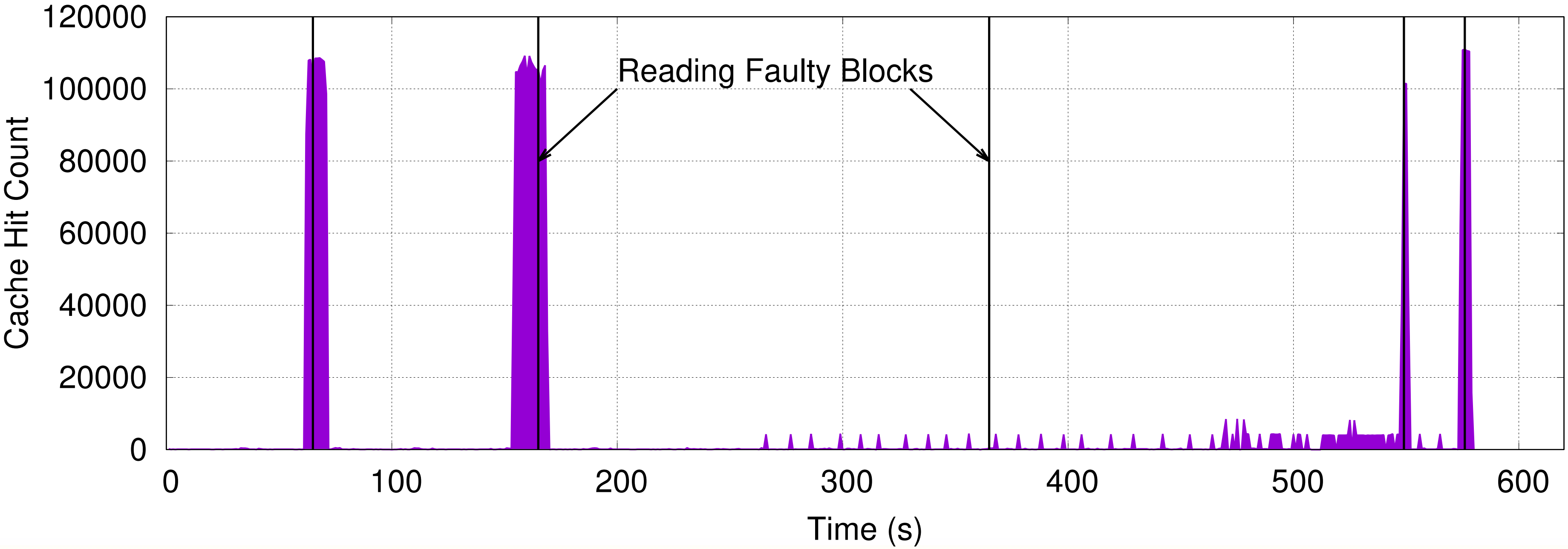
# Are Disk Write Errors Captured?



Potential weakness to detect disk write errors!

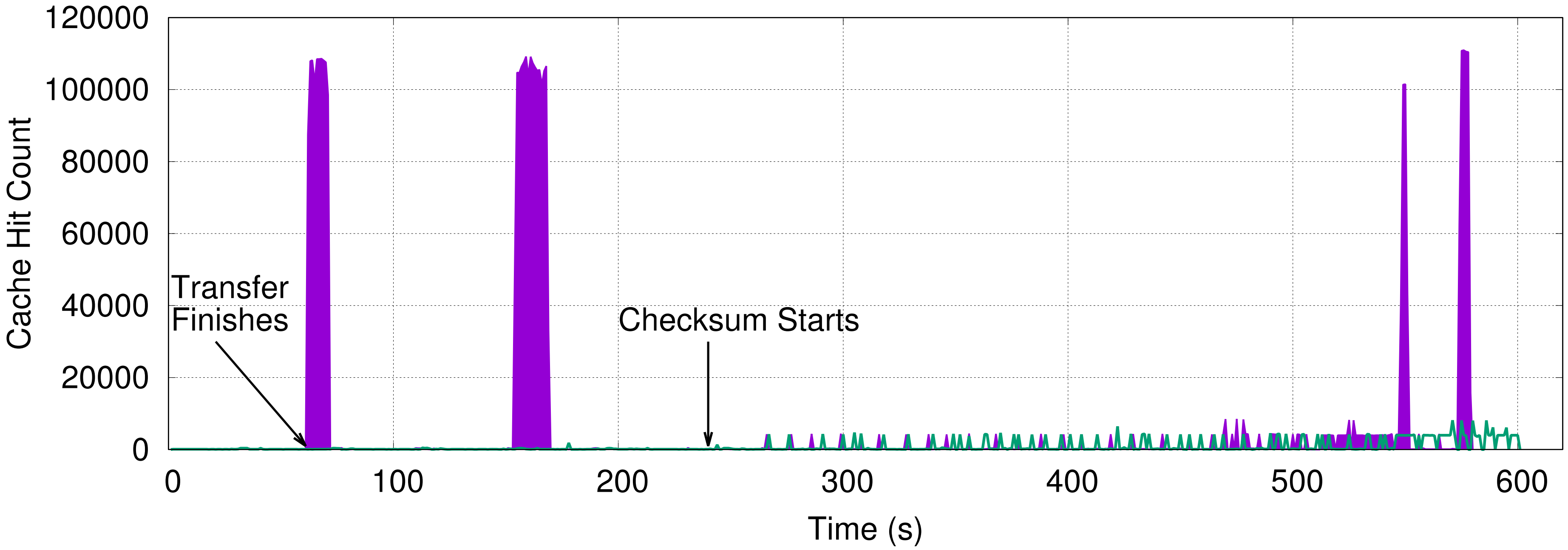
# Testing Integrity Verification Against Faults

- Four files 1-5 GB and one file 24 GB. Memory size is 20 GB
- One fault injected for each file during disk write
- Traditional approach failed to catch 4 out of 5 faults!



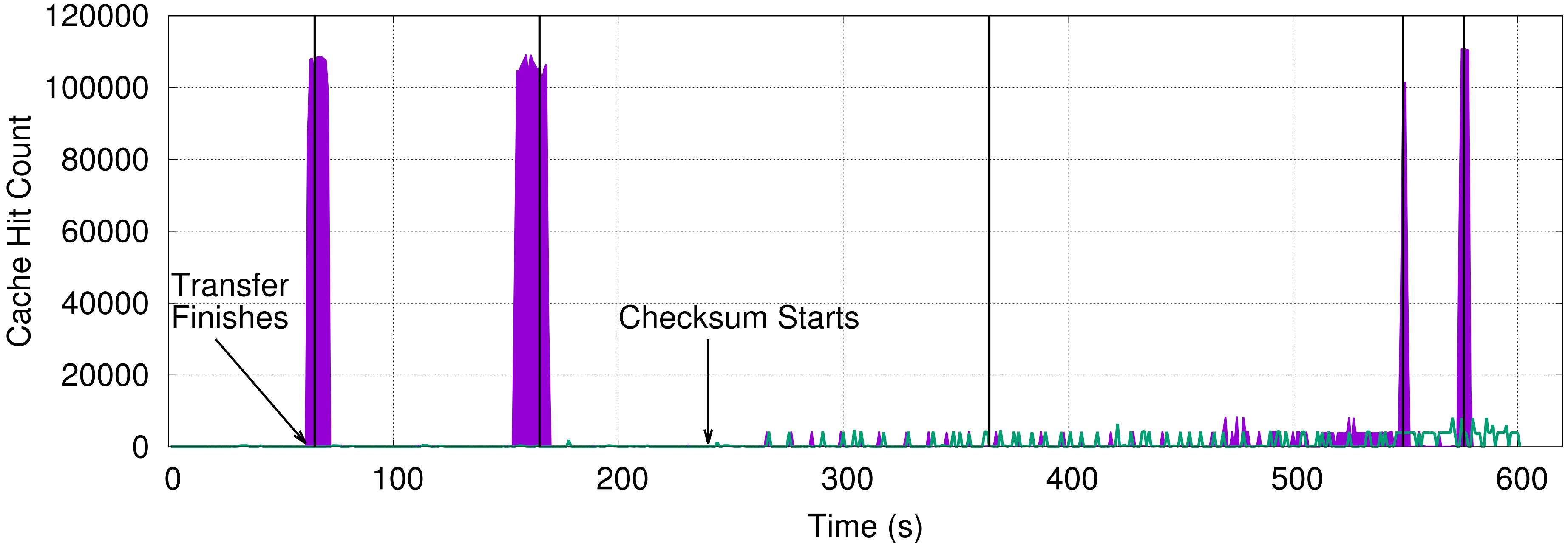
# Proposed Solution

- **Secure Integrity Verification Algorithm (SIVA)**
  - Delay checksum computation to let kernel remove files from cache
  - Ensures that files are read from disk



# Proposed Solution

- **Secure Integrity Verification Algorithm (SIVA)**
  - Delay checksum computation to let kernel remove files from cache
  - Ensures that files are read from disk



# Future Work

- SIVA leads to ~4% cache hits. Can we reduce it even lower to avoid missing any disk corruptions?
- Delaying checksum incurs execution time overhead in return of stronger fault tolerance. How to optimize execution time without sacrificing accuracy?
- Explore ways to detect file cache removal to start checksum earlier

Questions?