Gridftp Data Integrity Verification

Draft proposal, v1.0

Timur Perelmutov, FNAL

This document addresses the issues of the Data Integrity Verification as it was proposed in the GridFTP Protocol Improvements document (https://forge.gridforum.org/projects/gridftp-wg/document/GridFTP_v1.0_Improvements).

I propose to add 2 new commands “CHKSM” and “CHKSMOF” which would allow a checksum to be sent/received for STO/ESTO/RETR/ERET transfers.

I also propose to extend the Extended Block Header format with the checksum information for each block.

The detection of data corruption becomes possible on a block-to-block basis once the checksum is included with each block. Therefore, I also propose the addition of a recovery mechanism that would allow retransmission of the corrupted block.

CHKSM command

Command syntax is
CHKSM <checksum type> <checksum value as a hexadecimal number>

The checksum type is a hexadecimal constant with the same value as the value of the “Checksum Type “ field, as described in Extended Block Modification section. Another possibility is to use well defined string constants as <checksum type> , for example “crc”, “adler32”, etc.

The purpose of the checksum command is to communicate the checksum of the data transferred by the first STOR / ESTO following issue of CHKSM . The transfer to the gridftp server can be in either stream or extended block mode and can be either full or partial. The server can use the information transmitted by the CHKSM command for immediate data integrity verification, for storage in transferred file metadata, etc. In case of success the server responses with the return code 200.

CHKSOF command

Command syntax is
CHKSOF <checksum type> <path> [<offset> <length>]

Note: offset and legth fields are optional

The purpose of the CHKSOF command is to allow the client to discover the checksum of the full / partial file data prior to receiving the this data via RETR / ERET and then use this information for performing the verification of the received data integrity. The successfull response to the command is the return code 200 followed by checksum as a hexadecimal number.

Extended Block Modification.

The purpose of the modification is to allow the receiving part of the eblock mode transfer to validate the integrity of each extended block.

The new Extended Block Header Format is

Filed #	Name	Length
field 1	Descriptor	8 bits
field 2	Byte count	64 bits
field 3	Offset Count	64 bits
field 4	Checksum Type	16 bits
field 5	Checksum Length	16 bits
field 6	Checksum	As specified by field 5

The three new fields are “Checksum Type”, “Checksum Length” and “Checksum”,

The value of “Checksum” field is one of the following constants

Constant Name	Value
NOCHECKSUM	0 # is used for example if no data is transmitted by the block
CRC	1
ADLER32	2

Note: more types of checksums or digital signatures can be added as will be seen fit by gridftp working group.

The value of “Checksum Length” field is the length of checksum in bytes or 0 if the value of “Checksum Type” is NOCHECKSUM.

One possible variant of this Extended Block modification is to start with the data immedialely after field 5 and append the checksum at the end of the block. In this case the checksum would be of the both the header and the data parts of the block and would allow one to verify integrity of the both parts.

Extended block corruption recovery.

Note: This mechanism described here is inspired by the concepts described in the

document “Modification of Extended Block Mode Protocol” (https://forge.gridforum.org/projects/gridftp-wg/document/XBlockDraft/en/1). If it is accepted then this document will need to include the xblock recovery mechanism.

Purpose: Recovery from the extended block data corruption.
Extended block corruption can be signaled to the sender of the block by the receiver using the stream with the opposite to data flow direction of same data socket, on which the corrupt block was transmitted. The corruption notification could be of the format:

RESEND <offset> <length>
where <offset> and <length> are hexadecimal integers identical to the values of “Offset Count? and ?Byte Count? fields of the header the of the corrupt block transmitted.

In the situation where corruption is detected but recovery is not possible the server has an option of returning the failure response to client via the control chanel.

A command for discovery of checksum types supported by the server can be added as well.