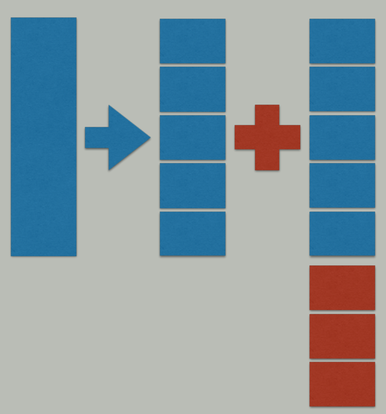
Erasure-Coded File Distribution in DIRAC

Sam Skipsey[1], Dan Protopopescu[1], Paulin Todev[1], David Britton[1], Gareth Roy[1], David Crooks[1]

[1]University of Glasgow

Background

Traditional Grid storage mechanisms use whole file replication for data resilience and availability.
The number of supported replicas has been reducing over time, as space becomes increasingly contended.
One way out is to change paradigms, and switch to Erasure Coded striping as a T2 data distribution mechanism.

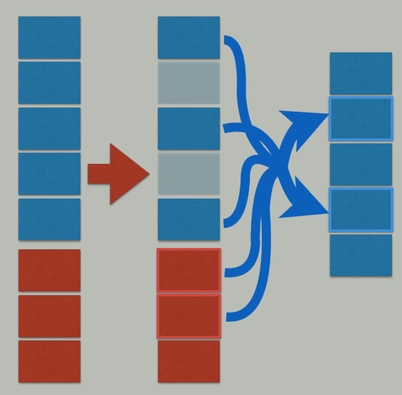


File placement for Erasure-coded files:

- Split file into k chunks (padding last chunk if needed).
- Generate m additional chunks via linear EC mapping.
- Distributed m+k chunks across filesystem (Grid)

File recovery for Erasure-coded files:

- Recover at least k chunks from filesystem (Grid)
- (Lost primary chunks reconstructed using coding chunks)
- Concatenate primary chunks to destination file.



DIRAC

The LHCb DIRAC system provides a file catalogue layer (DFC), with extensible metadata. This can be used as the basis of an object striping layer, as well as the mundane uses it was intended for. The API language is python.

ZFEC

The ZFEC open-source Reed-Solomon erasure code implementation provides a python API. This is freely available and not patent encumbered (and is part of the Tahoe-LAFS filesystem).

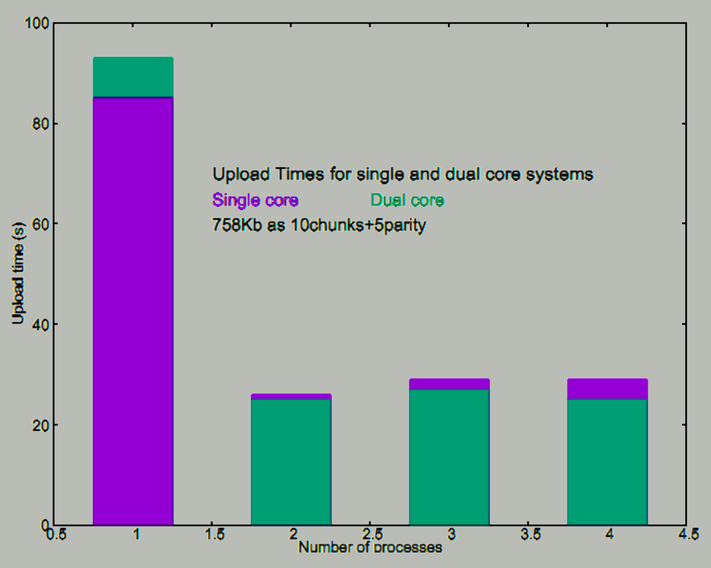
https://tahoe-lafs.org/

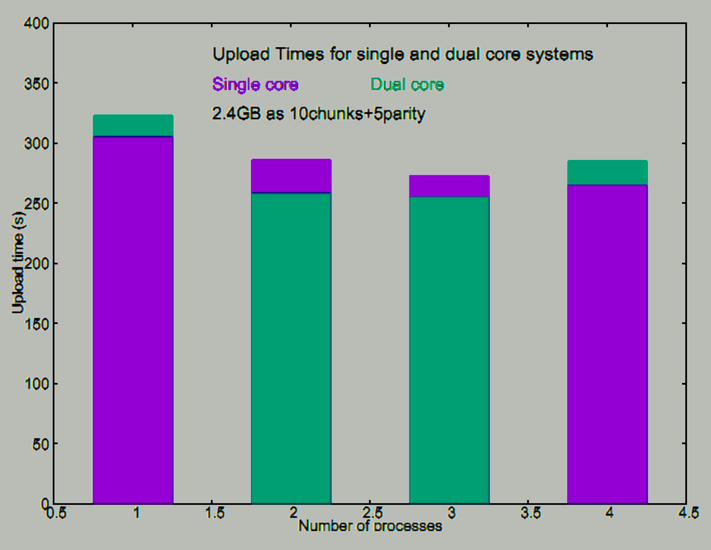
Implementation

Erasure Coded files are implemented as directories (with the original file name) in DFC space, containing the logical names of all of the distributed chunks. Chunks are distributed across the vector of known SE endpoints supporting the given VO. Chunk metadata is encoded in the chunk filename (in standard zfec format), and also in DFC metadata tags applied to the logical entry ('EC_FILE' (name), 'EC_VERSION', 'TOTAL' (number of files), 'SPLIT' (total number of original chunks)).

Benchmarks

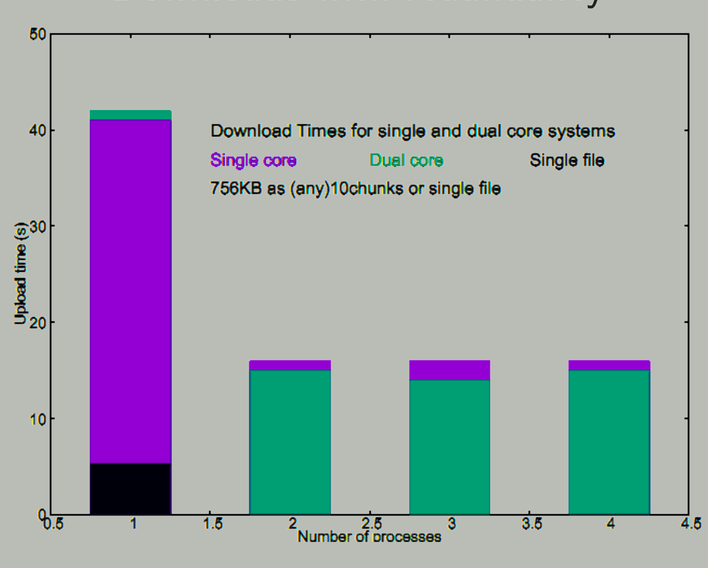
Uploads with redundancy

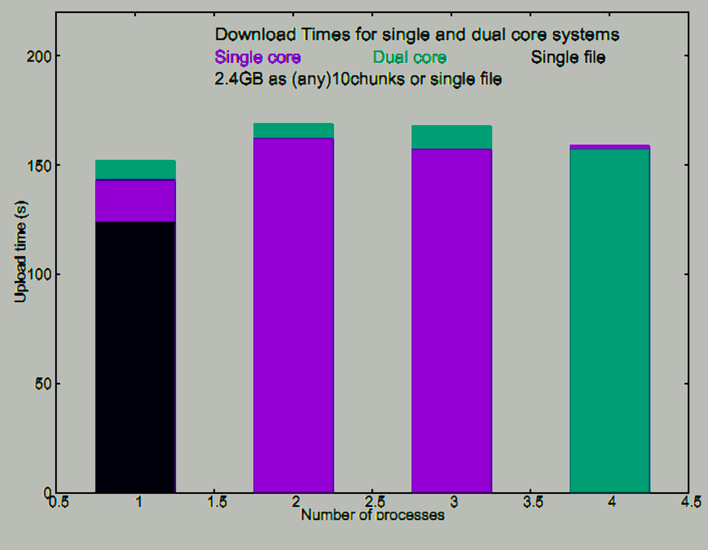




We can take advantage of the chunk distribution by performing parallel transfer across all chunks when putting and getting files.
This helps to compensate for the time overhead of generating the encoding chunks. For small files, the overhead of additional communication with the DFC (and recoding overhead) is significant, versus a single complete copy.

Downloads with redundancy





Comments

It appears that the DFC metadata namespace is global, resulting in our adhoc metadata tags being visible to other users of the DFC. It is not clear if this is desirable.

https://github.com/ptodev/Distributed-Resilient-Storage/ and https://github.com/aoanla/Distributed-Resilient-Storage