BLUESTORE INTERNALS

SMALL WRITE STRATEGIES

- *U*: Uncompressed write of a complete, new blob.
  - write to new blob
  - kv commit
- *P*: Uncompressed partial write to unused region of an existing blob.
  - write to unused chunk(s) of existing blob
  - kv commit
- *W*: WAL overwrite: commit intent to overwrite, then overwrite async. Must be chunk_size = MAX(block_size, csum_block_size) aligned.
  - kv commit
  - wal overwrite (chunk-aligned) of existing blob
- *N*: Uncompressed partial write to a new blob. Initially sparsely utilized. Future writes will either be *P* or *W*.
  - write into a new (sparse) blob
  - kv commit
- *R+W*: Read partial chunk, then to WAL overwrite.
  - read (out to chunk boundaries)
  - kv commit
  - wal overwrite (chunk-aligned) of existing blob
- *C*: Compress data, write to new blob.
  - compress and write to new blob
  - kv commit

POSSIBLE FUTURE MODES

- *F*: Fragment lextent space by writing small piece of data into a piecemeal blob (that collects random, noncontiguous bits of data we need to write).
  - write to a piecemeal blob (min_alloc_size or larger, but we use just one block of it)
  - kv commit
- *X*: WAL read/modify/write on a single block (like legacy bluestore). No checksum.
  - kv commit
  - wal read/modify/write

MAPPING

This very roughly maps the type of write onto what we do when we encounter a given blob. In practice it's a bit more complicated since there might be several blobs to consider (e.g., we might be able to *W* into one or *P* into another), but it should communicate a rough idea of strategy.

	raw	raw (cached)	csum (4 KB)	csum (16 KB)	comp (128 KB)
128+ KB (over)write	U	U	U	U	C
64 KB (over)write	U	U	U	U	U or C
4 KB overwrite	W	P | W	P | W	P | R+W	P | N (F?)
100 byte overwrite	R+W	P | W	P | R+W	P | R+W	P | N (F?)
100 byte append	R+W	P | W	P | R+W	P | R+W	P | N (F?)
4 KB clone overwrite	P | N	P | N	P | N	P | N	N (F?)
100 byte clone overwrite	P | N	P | N	P | N	P | N	N (F?)