Why Reading Patterns Matter
in Storage Coding & Scheduling Design

Ulric J. Ferner
Massachusetts Institute of Technology
Cambridge, MA USA
uferner@mit.edu

Emina Soljanin
Bell Labs, Alcatel-Lucent
Murray Hill, NJ USA
emina@research.bell-labs.com

Muriel Médard
Massachusetts Institute of Technology
Cambridge, MA USA
medard@mit.edu

Abstract—Coding techniques for storage systems are gaining traction in data center (DC) applications, owing to their data survivability performance, and more recently, to their ability to mitigate traffic congestion. This paper considers stochastic allocation schedules in networks that admit bulk file requests, across three drive blocking models. We consider a block-based code and a stochastic scheduling algorithm which is beneficial in the case of continuous chunk read patterns. In particular, we demonstrate that in systems with continuous chunk reading patterns, when drive blocking is either independent or from traffic congestion, block coded storage can reduce average download time by 10–66%, given modern system parameters. However, a distinction should be made between systems with continuous and those with interrupted chunk read patterns. For interrupted chunk read systems, given our allocation algorithm that performs well for continuous reads, block coded storage performance can be worse than replication; numerical illustrations show relative losses over 66%. These illustrations demonstrate that to harness the full benefits of coded storage and to avoid pitfalls, careful attention must be paid to continuous vs. interrupted chunk reading patterns, codes other than block codes should be considered, as could joint code-scheduling design.

Keywords—Blocking probability; bulk requests; cloud computing architectures; coded storage; design patterns; download time; file chunks; reading patterns; storage.

I. INTRODUCTION

Increasing demand for cloud computing and data center (DC) storage continues worldwide unabated. To increase the number of user data streams that can be simultaneously managed, DCs commonly divide, stripe, or chunk large files across multiple drives to speed up read times, and entire files or file chunks are replicated throughout and across DCs.

Coding techniques such as block maximum distance separable (MDS) codes and rateless codes are gaining traction in enterprise DC applications, owing to their data survivability performance [1], and more recently, to their ability to mitigate against traffic congestion [2], [3]. Traffic congestion is caused by content being temporarily unavailable to read owing to other users concurrently overwhelming DC resources.

We consider a network of drives, where each drive stores file chunks (file chunks are fixed-size subsets of some file). Drives read out chunks in response to incoming read requests, and a central scheduler allocates read requests to drives. A number of metrics can be used to understand traffic congestion in such networks, such as blocking probability and queueing delay. Blocking probability captures the finite I/O of modern drives, be they hard disk drives or solid state caches, and can be used to inform a wide variety of design decisions. It has also been used to show that for initial chunk requests, coded storage scheduling options can be larger than equivalent uncoded systems [2]. This paper builds upon these results and analyzes systems that admit bulk requests. In particular, the main contributions of this paper are as follows.

- We present a system model that admits a variety of drive blocking model components, and use as our performance metric the average delay for bulk request downloads; we analyze drives with blocking from traffic congestion, and build up to this by first considering drives without blocking, and then drives with traffic independent blocking.
- We consider a block-based code and stochastic scheduling algorithm which is beneficial given continuous chunk reading patterns. In such systems, when blocking is independent of traffic or from traffic congestion, block coded storage can reduce average download time 10–66%, given modern storage parameters. (Continuous and interrupted chunk reading patterns are defined in Sec. III-C.)
- We conclude that there is a distinction between systems with continuous and those with interrupted chunk reading patterns. Perhaps counter-intuitively, for interrupted chunk reading patterns, given our allocation algorithm that performs well for continuous reads, block coded storage performance can be worse than replication; numerical illustrations show relative losses over 60%. Intuitively, in block coded storage it is easier to start a new read, but harder to finish an incomplete one.

Techniques that are have been proposed and are starting to be used to mitigate data center traffic congestion, without coding, include generating redundant requests that are sent to...
a group of drives. For example, Vulimiri et al. [4] proposes replicating file read requests and randomly sending those requests to more drives than the minimum number required. Using coding, work closest to this paper is the queueing theoretic analysis of the benefits of block coding in DCs through fork-join queues [5], although scheduling is not considered. Queueing delay in systems with deterministic scheduling with perfect drive-state information is considered in [6]. To the authors’ knowledge, this is the first paper to consider stochastic scheduling when coding is used to mitigate traffic blocking in distributed storage systems. This paper does not aim to develop optimal scheduling algorithms for block coded storage, only to analyze reasonable scheduling algorithms for continuous reading patterns, that provide insight into the benefits and pitfalls of coded storage.

The remainder of this paper is organized as follows. Sec. II describes the system model and download algorithm. Sec. III details the analysis for the three blocking models (results for the most general model, with traffic congestion and interrupted chunk reads, are described in Sec. III-C). Sec. IV discusses results, and finally Sec. V presents conclusions.

II. FILE STORAGE MODEL AND DOWNLOAD ALGORITHM

A file of $T$ chunks, where each chunk is a fixed-size subset of some file, is stored within $WT$ chunks of storage space; $W \in \mathbb{N}_+$. A file is downloaded when sufficient chunks have been read from drives so that all $T$ chunks from the file can be decoded. Time is slotted in rounds, where in each round a chunk is selected randomly with replacement. Random scheduling of requests like this is one way of load balancing a network, and modern networking equipment can often be easily configured to do so [7]. Key modeling parameters and assumptions in this paper are:

- **Uncoded storage design**: In the uncoded or replication system, each chunk is replicated $W$ times.
- **Block coded storage design**: In the coded storage system, we use a single MDS block code, where the $T$ file chunks are mapped into $WT$ coded chunks such that any $T$ out of $WT$ are sufficient for decoding.
- **Chunk reading patterns**: Reading a chunk requires $l \in \mathbb{N}_+$ rounds at a drive storing that chunk. We distinguish between systems with continuous and interrupted chunk reading patterns where $l = 1$ and $l > 1$, respectively.

We assume that storage is distributed in the sense that each chunk is stored on a different drive. A chunk on a drive can be read simultaneously up to $K$ users, where $K$ is a function of the drive’s I/O access bandwidth.

We model user requests for content as requests for the entire file (that is, bulk requests for its chunks), which arrive in the system via a Poisson process at rate $\lambda$. (The Poisson nature of arrivals is assumed to simplify analysis.) Upon arrival, each file request is turned into a number of sequential individual chunk requests. In this manuscript, we consider three blocking models for drives to respond to service requests:

1) **No blocking**: There is enough I/O bandwidth to serve each request immediately. This model is used as a reference point and to build intuition around the full problem.

2) **Independent blocking**: Each request is blocked with some probability, independent of other requests. This model is appropriate when there is enough I/O bandwidth to serve each request immediately, but chunks can be dropped with some probability in transport, or multiple schedulers are competing for the same drives.

3) **Traffic congestion**: The I/O bandwidth is finite and each chunk can be read simultaneously by $K$ users (one per each I/O access bandwidth slot). Thus a request is blocked if there are already $K$ being served.

Because of blocking, some chunks will have to be requested repeatedly in order to download the file. We are interested in evaluating the file download latency as measured by the number of chunk requests, or equivalently the number of scheduling rounds, $N \geq T$. We are particularly interested in how the number of requests or rounds $N$ concentrates around its average value $E[N]$. Knowing these quantities and behavior is of interest for resource provisioning, e.g., if we provide $E[N]$ systems each operating as described above, we will know how likely we are to immediately satisfy a file request.

We are interested in the number of requests in the uncoded system, where we refer to it as $N^u$, and the block coded system, where we refer to it as $N^c$. The file download follows the process outlined as Algorithm 1. Decoding time is not considered in Algorithm 1 or in this paper’s analysis. Generally, the overhead of decoding chunks from coded files can be managed so as to not interfere with other required processing, such as video codec decoding. For more details, see manuscripts [2] [8], and the references therein.

It is important to understand how the algorithm step in Line 2 is carried out depending on whether uncoded or block coded storage is used. When block coding is used, one of the $TW$ coded chunks is selected according to some probability distribution $p^c$ (independently of the previous selections). Note that $p^c$ has the support $\{1, \ldots, TW\}$ and there is only a single drive storing the selected chunk. On the other hand, when no coding is used, one of the $T$ original file chunks is selected according to some probability distribution $p^u$ (independently of the previous selections). Note that $p^u$ has the support $\{1, \ldots, T\}$ and there are $W$ drives storing the selected chunk. Therefore the dynamics of reads requested of drives depends on whether or not coding is used, and will be reflected in our traffic congestion model.

Remark: Note that the probability that a request for one of the $TW$ coded chunks is blocked in the block coded system may be larger than the probability that one of the $T$ file
Algorithm 1 File Download (upon a file request arrival)

1: Set the chunk request counter to 0:
   \( N = 0 \)
2: Pick a (coded) file chunk, and request it from its drive(s)
3: Increment the chunk request counter:
   \( N \leftarrow N + 1 \)
4: if the access to the chunk is blocked then
5:   Go to Line 2
6: else
7:   Cache individual chunk
8: if \( T \) different (coded) chunks have been cached then
9:   END/EXIT
10: else
11:   Go to Line 2
12: end if
13: end if

chunks is blocked in the uncoded system. However, the file download with MDS block coding requires collecting any \( T \) out of \( TW \) block coded chunks whereas the file download without coding requires collecting one chunk from each of the \( T \) \( W \)-replicated groups, and it is in this flexibility that the advantage of using coding resides. It is instructive to observe the similarity with physical layer scenarios, where introducing coding increases the raw error rate for the same \( E_b/N_0 \) but coding still offers overall gains by correcting some errors.

III. ANALYSIS

In this section we determine the expected value of the number of requests in Algorithm 1, for each of our three blocking models used in Line 4. Unless otherwise specified, we assume random chunk selection with uniform distribution \( u \), and support depending on whether coding is used or not.

A. No Blocking

Consider the case when there is no drive blocking. In the uncoded scheme, let \( N^u(u) \) be the number of algorithm rounds required to have collected at least one copy of all \( T \) chunks. Since each chunk is selected uniformly at random and independently from previous selections, this scenario is equivalent to the classic coupon collector problem [9], in which the expected number of rounds is given by

\[
E[N^u(u)] = TH_T
\]

where \( H_T \) is a harmonic number given by

\[
H_T = \sum_{i=1}^{T} \frac{1}{i} = \log T + \gamma + \frac{1}{2T} + O(T^{-2}), \tag{2}
\]

and \( \gamma \) is the Euler’s constant.

In the coding scheme, let \( N^c(u) \) be the number of reads required for a single user to decode, i.e., the number of reads required to receive \( T \) different block coded chunks (note that each coded chunk is unique). Let \( t_i^c \) be the number of rounds required to read a previously unread coded chunk after having cached some \( i - 1 \). We have

\[
N^c(u) = \sum_{i=1}^{T} t_i^c
\]

where \( t_i^c \) is a geometric random variable with the probability of success \( (TW - (i - 1))/TW \). The expected number of rounds is thus given by

\[
E[N^c(u)] = \sum_{i=1}^{T} E[t_i^c] = \frac{TW}{TW} + \frac{TW - 1}{TW} + \cdots + \frac{TW - T}{TW}
\]

\[
= TW(H_{TW} - H_{(TW-1)}). \tag{3}
\]

B. Independent Blocking

Suppose that drive \( i \) has blocking probability \( p_b(i) \). Drive blocking is traffic independent and also independent across chunks and time slots. This moves us from the prior model, without blocking, to one that is closer to the full model in which traffic congestion causes drive blocking. This particular model is appropriate when there is enough I/O bandwidth to serve each request immediately, but chunks can be dropped with some probability in transport, or multiple schedulers may be competing for the same drives. No assumption is made about the uniformity of chunk replication. In this system, the expected number of reads should increase. In this subsection we assume each drive stores only a single chunk. We will unwind this assumption in the subsection on traffic congestion.

Given independent blocking, let \( N_{ind}^u \) and \( N_{ind}^c \) be the number of rounds required to have collected at least one copy of all \( T \) chunks in the uncoded and block coded systems, respectively.

The probability that replicas of any particular chunk \( f_i \) or \( f_i^c \) are all blocked is given by \( \prod_{i \in \mathcal{M}_f} p_b(i) \) and \( p_b(i) \) for the replication and coding schemes, respectively, where \( \mathcal{M}_f \) is the set of drives storing chunk \( f_i \). Consider the case with symmetric blocking probability across chunks, i.e., \( p_b = p_b(i), \forall i \). Then the expected number of rounds is

\[
E[N_{ind}^u(u)] = \mathcal{P}_{ind}^{u} TH_T \tag{4}
\]

and

\[
E[N_{ind}^c(u)] = \mathcal{P}_{ind}^{c} TW(H_{TW} - H_{(TW-1)}) \tag{5}
\]

for the uncoded and block coded schemes, respectively, where \( \mathcal{P}_{ind}^{u} \) and \( \mathcal{P}_{ind}^{c} \) are blocking probability penalties given by

\[
\mathcal{P}_{ind}^{u} = (1 - p_b)^{-W} \tag{6}
\]

and

\[
\mathcal{P}_{ind}^{c} = (1 - p_b)^{-1}. \tag{7}
\]
respectively.

We now extend our analysis of the independent blocking to systems with non-uniform allocation algorithms, in which we consider general random allocation algorithms with arbitrary \( p_a(i) \)'s and \( p'_a(i) \)'s. This could occur if, for instance, chunks are not replicated in the same quantities throughout the system. We first solve the uncoded, and then the block coded scheme.

We apply the Newmann-Shepp symbolic method to solve both the uncoded and block coded schemes, the derivation of which we summarize here. It has been shown that the Newmann-Shepp symbolic method in many cases reduces the computational complexity in numerically solving the derived indefinite integrals [9]. For a full description of this technique and related tools, we refer the interested reader to [10] and [9]. For the uncoded scheme, represent the set of possible outcomes after \( n \) successful, i.e., non-blocked, reads by the symbolic polynomial

\[
(f_1 + \cdots + f_T)^n
\]  

where \( f_i \) is the \( i \)th uncoded chunk and, for example, symbol \( f_i^2 \) represents the outcome of chunk \( f_i \) being read exactly twice. Let \( \mathcal{E} \) be the event that \( T \) chunks are read, or equivalently that there are at least \( T \) unique chunk indices in the resulting set of reads. Then let \( \mathcal{J} \) be an operator that removes terms that correspond to \( \mathcal{E} \). The expected number of rounds until a user can decode is then

\[
E[N_u^{\text{ind}}(p)] = \mathcal{P}_u^{\text{ind}} \sum_{n=0}^{\infty} P(N^u > n)
\]

\[
= \mathcal{P}_u^{\text{ind}} \sum_{n=0}^{\infty} \mathcal{J} \left( (f_1 + \cdots + f_T)^n \right) \bigg|_{f_i=p_a(i)}
\]

where

\[
\mathcal{J} \left( (f_1 + \cdots + f_T)^n \right) \bigg|_{f_i=p_a(i)}
\]

can be interpreted as the probability of an \( n \)-read failure in the uncoded system, where we have an insufficient number of unique chunks. We cast our system as a generalized birthday problem [9], asking what is number of rounds required to read at least \( T \) unique chunks from the set, where reading is done randomly with replacement.

Using the equality,

\[
\int_0^{\infty} t^n \frac{e^{-t}}{n!} dt = 1,
\]

we combine (11) with (9), giving

\[
E[N_u^{\text{ind}}(p)] = \mathcal{P}_u^{\text{ind}} \sum_{n=0}^{\infty} \int_0^{\infty} e^{-t} \mathcal{J} \left( (f_1 + \cdots + f_T)^n \right) \bigg|_{f_i=p_a(i)} dt
\]

\[
= \mathcal{P}_u^{\text{ind}} \int_0^{\infty} e^{-t} \mathcal{J} \left( \prod_{i=1}^T e^{f_i^t} \right) dt \bigg|_{f_i=p_a(i)}
\]

where \( \mathcal{J} \) is a linear operator that removes terms containing all chunk indices. We represent a term containing chunk \( f_i \) as \( e^{f_i^t} - 1 \), and the \( \mathcal{J} \) operator factor simplifies to

\[
\mathcal{J} \left( \prod_{i=1}^T e^{f_i^t} \right) = \sum_{S \in \mathcal{P}_{\leq T}(1, \ldots, T)} \prod_{i \in S} (e^{f_i^t} - 1)
\]

(13)

where \( \mathcal{P}_{\leq T}(1, \ldots, T) \) is the powerset of all chunk indices containing less than \( T \) indices. \( E[N_u^{\text{ind}}(p)] \) can be further simplified using the property \(-t = -t \sum_i p_a(i)\), giving

\[
E[N_u^{\text{ind}}(p)] = \mathcal{P}_u^{\text{ind}} \int_0^{\infty} \left( 1 - \prod_{i=1}^T (1 - e^{-p_a(i)t}) \right) dt.
\]

(14)

Equation (14) can be quickly numerically evaluated. If all chunks have the same allocation probability \( p_a(i) = 1/T \), \( E[N_u^{\text{ind}}(p)] \) simplifies further to

\[
E[N_u^{\text{ind}}(p)] = \mathcal{P}_u^{\text{ind}} \int_0^{\infty} \left( 1 - (1 - e^{-t/T})^T \right) dt
\]

(15)

which we numerically evaluated for various \( T \) values and matched (4) in all attempted cases.

We now contrast the uncoded and block coded schemes for non-uniform allocation algorithms. Single block coding is similar to replication except that, crucially, we have \( TW \) unique chunks to read from without replication and only require any \( T \) unique chunks. The expected number of coded reads \( E[N_c^{\text{ind}}(p)] \) is then given by

\[
E[N_c^{\text{ind}}(p)] = \mathcal{P}_c^{\text{ind}} \int_0^{\infty} e^{-t} \mathcal{J} \left( \prod_{i=1}^{TW} e^{f_i^t} \right) dt \bigg|_{f_i=p_a(i)}
\]

(16)

Operator \( \mathcal{J} \) removes terms that contain at least \( T \) unique chunks, giving

\[
\mathcal{J} \left( \prod_{i=1}^{TW} e^{f_i^t} \right) = e^{(f_{1}^t + \cdots + f_{TW}^t)} - \sum_{S \in \mathcal{P}_{\geq T}(1, \ldots, TW)} \prod_{i \in S} (e^{f_i^t} - 1)
\]

(17)

where \( \mathcal{P}_{\geq T} \) is the powerset of groups greater than or equal to \( T \) indices. We use the notation \([u^r]f(u)\) for the coefficient of \( u^r \) in the power series development of function \( f(u) \), and rewrite the coding \( \mathcal{J} \) operator term more compactly, without using powersets or factorials, as

\[
\mathcal{J} \left( \prod_{i=1}^{TW} e^{f_i^t} \right) = \sum_{r=0}^{TW} [u^r] \prod_{i=1}^{TW} (1 + u(e^{f_i^t} - 1))
\]

(18)
giving

\[ E[N_{\text{ind}}^c(p)] = \mathcal{P}_{\text{ind}}^c \int_0^\infty e^{-t} \sum_{i=0}^{T-1} \prod_{i=1}^{TW} \left(1 + u(e^{p_a(i)} - 1)\right) dt. \]  

(19)

If all chunks have the same allocation probability \( p_a(i) = 1/(TW) \), \( \forall i \) then \( E[N_{\text{ind}}^c] \) simplifies further to

\[ E[N_{\text{ind}}^c(u)] = \mathcal{P}_{\text{ind}}^c \int_0^\infty e^{-t} \sum_{r=0}^{T-1} (TW)^r (e^{\lambda/(TW)} - 1)^r dt, \]  

(20)

which we numerically evaluated for various \( (T, W) \) values and found to match (5).

To illustrate numerical results and to highlight potential advantages of block coded over uncoded systems, we consider uncoded allocation probability density functions that are both uniform as well as those that follow a power-law. If a nonuniform allocation distribution exists, we assume it is caused by unequal chunk replication in the system. As a power-law distribution, we select the Zipf allocation algorithm, commonly used to model web-based content demands [11], [12]. The Zipf distribution gives allocation probabilities of

\[ p_a(i; \alpha, T) = \frac{i^{-(\alpha+1)}}{H_T^{(\alpha+1)}}, \]  

(21)

and

\[ p_a(i; \alpha, TW) = \frac{i^{-(\alpha+1)}}{H_{TW}^{(\alpha+1)}}, \]  

(22)

across \( T \) and \( TW \) chunks for the uncoded and block coded systems, respectively, and function \( H_T^{(\alpha)} \) is the generalized harmonic number given by

\[ H_T^{(\alpha)} = \sum_{k=1}^{T} \frac{1}{k^{\alpha}}. \]  

(23)

We set \( \alpha = 0.6 \), similar to [11], [12]. In the equivalent block coded system, since all chunks are coded together, we compare the comparative block coded system having a uniform allocation distribution as well as a Zipf distribution. See Table I for select numerical examples. All integrals were solved using the Newton-Cotes method with eight points. Even at low \( T \) values, it hints at potentially large gains in the average number of rounds required, motivating full exploration of the traffic congestion model.

C. Traffic Congestion

Consider traffic congestion which causes temporary data unavailability, in which the number of available I/O slots for any chunk set is given as a function of the number of requests that chunk set is currently serving. This is the most realistic and complex blocking model considered in the paper. We first consider systems in which drives store chunks that can be read in a single scheduling round or continuously, so \( l = 1 \), and then extend analysis to systems in which drives store multi-round chunks, or chunk reading patterns are interrupted and take multiple scheduling rounds, i.e., \( l \in \mathbb{N}_+ \).

1) Continuous Chunk Reading Patterns: As has been done thus far throughout the paper, we set \( l = 1 \) in this subsubsection. In the uncoded scheme, we model the probability of blocking for each chunk as follows. As a reminder, there are \( W \) copies of each chunk in the subsystem, and each drive has \( K \) I/O slots, with average slot service rate \( \mu \). Given equal demand across chunks, drives that store the same chunk have a Poisson arrival rate of \( \lambda/T \) to each chunk set, and there are \( KW \) I/O slots available in total to service that chunk. The blocking probability for each chunk is then given by

\[ \frac{\rho^{KW}/(KW)!}{\sum_{i=0}^{KW} \rho^i/i!}, \]  

(24)

where \( \rho = \lambda/(\mu T) \).

Consider the single block coded scheme with uniform chunk layout. We model the probability of blocking for a selected coded chunk similarly to the uncoded scheme. In particular, each drive still has \( K \) I/O slots, and chunks are serviced at average rate \( \mu \) in each slot. Since each block coded chunk is unique, each has a Poisson arrival rate of \( \lambda/(TW) \) and only \( K \) I/O slots available to it. The blocking probability of each block coded chunk is then given by

\[ \frac{\rho^K/(WK)!}{\sum_{i=0}^{KW} \rho^i/(W^i i!)}. \]  

(25)

This gives blocking probability penalties of

\[ \mathcal{P}_{\text{traffic}} = \left[ 1 - \left( \frac{\rho^{KW}/(KW)!}{\sum_{i=0}^{KW} \rho^i/i!} \right)^W \right]^{-1}. \]  

(26)
reads from chunks instead of larger individual reads. We assume $2 \cdot 2^\rho$ unique chunks, each with at least $2 \cdot 0^\rho$ reads. The techniques that are used in this section can also be directly used to analyze systems in which drives store $l$ chunks instead of larger individual chunks.

To complete reading a chunk a scheduler requires $l$ reads from copies of that chunk, i.e., a file is downloaded when we have collected at least $l$ reads from $T$ chunks. We assume that if a user accesses the same chunk or a copy of that chunk in different rounds, that in each round they will receive usable information until the chunk is completed. Crucially, in this model information from a partially read chunk cannot be mixed with that of another partially read chunk.

Such a system is a simple extension of single-round chunk systems if we use the Newmann Shepp symbolic method. Consider the uncoded scheme. We revisit and update (12) for traffic congestion giving,

$$E[N_{\text{traffic}}^u(p)] = \mathcal{P}_{\text{traffic}}^u \int_0^\infty e^{-t} \mathcal{J}_l \left( \prod_{i=1}^T e^{f_{i,t}} \right) dt \bigg|_{f_i=p_u(i)}$$

and in this case the $\mathcal{J}_l$ operator term removes terms that contain at least $T$ unique chunks, each with at least $l$ reads. Allowing minor abuse of notation, define $e_{i,t}$ as the incomplete exponential function,

$$e_{i,t} = \sum_{i=0}^l t^i/i!$$

and rewrite the $\mathcal{J}_l$ operator factor as

$$\mathcal{J}_l \left( \prod_{i=1}^T e^{f_{i,t}} \right) = e^{(f_{1,t} + \cdots + f_{T,t})} - \prod_{i=1}^T (e^{f_{i,t}} - e_{i-1,t}^{f_{i,t}})$$

where the term $(e^{f_{i,t}} - e_{i-1,t}^{f_{i,t}})$ can be interpreted as a receiving at least $l$ reads of $f_i$. This gives

$$E[N_{\text{traffic}}^u(p)] = \mathcal{P}_{\text{traffic}}^u \int_0^\infty \left( 1 - e^{-t} \prod_{i=1}^T \left[ e^{p_u(i) t} - e_{i-1}^{p_u(i) t} \right] \right) dt$$

If the allocation algorithm is uniform $p_u(i) = 1/T \forall i$ then

$$E[N_{\text{traffic}}^u(u)] = \mathcal{P}_{\text{traffic}}^u \int_0^\infty \left( 1 - \left[ 1 - e_{i-1,i}^{f_{i,i} t} e^{-t/T} \right]^T \right) dt.$$

Using this machinery, the single block coded number of
The benefits of block coding are most pronounced when \( l = 1 \) (intermediate read). When \( l = 2 \), and use a uniform stochastic allocation, coding benefits are limited, although only in low replication- and low striping-number systems.

Figure 2. Given an uncoded system, an MDS block coded storage system, and our stochastic scheduling algorithm, this figure illustrates the effects of continuous and interrupted read patterns can have on comparative average download performance, as a function of the number of required scheduling rounds per chunk \( l \), and of replication-number \( W \). All computations are performed with \( \rho = 0.9 \), \( K = 2 \), and use a uniform stochastic allocation algorithm.

The models considered in this paper can be applied to physical networks constrained by finite I/O at storage drives, and where data requests to drives are managed across groups of drives. (A group can be managed by router, switch, load balancer or other networking equipment.) Such architectures then with independent blocking, with general stochastic allocation schedulers. (We do not seek to find optimal scheduling algorithms, but instead to find key insights on when single block coded storage may and may not improve system performance.)
are common in the Edge Layer within modern ‘fat tree’ data centers [13].

In the systems we considered there is a natural decoupling between combinatorial gains and blocking penalties, as in Algorithm 1. Results show that the blocking probability for single block coded systems can be larger than in uncoded systems, especially when replications numbers are 2-3 (common in modern systems).

For independent blocking systems with continuous reads, results demonstrate that the average number of scheduling rounds decreases in block coded, compared to uncoded systems. For traffic congestion systems, block coded storage benefits are most pronounced continuous reads are used, i.e., \( l = 1 \); the larger traffic congestion blocking penalty from block coding can be overcome by combinatorial gains.

In contrast, given a interrupted read system, an uncoded system scheduler can have a larger number of drive options when completing reads compared to a block coded system. In light of these trade-offs, it may be possible to design storage systems that use combinations or mixtures of replication and block coded storage (or replicas of blocks), as studied in communication systems [14].

It would be interesting to more deeply understand the trade-off when coding does and does not improve the average number of scheduling rounds to download files. It would also be insightful to understand the system engineering implications of interrupted reads on various applications, and to find optimal scheduling algorithms for coded storage systems.

V. CONCLUSIONS

In uncoded and block coded storage, with stochastic chunk allocation schedules, we have shown that single block coding can reduce the number of scheduling rounds for file downloads when chunk reading patterns cannot be interrupted. If chunk reading patterns can be interrupted and take multiple scheduling rounds, then it is not obvious if uncoded or block coded storage has superior performance, and is a function of replication number. Intuitively, in block coded storage it is easier to start a new read, but harder to finish an incomplete one. If the number of required scheduling rounds per chunk is high, block coded storage appears to increase, sometimes significantly, average download time. To harness the full benefits of block coded storage and to avoid pitfalls, this paper points to the careful attention that must be paid to block coded scheduling algorithms, and that coding is not necessarily a plug-and-play technique in a storage context.

ACKNOWLEDGMENT

This material is based upon work supported by Alcatel-Lucent under award #4800484399, the NetApp Faculty Fellowship, and an MIT Martin Family Society of Fellows for Sustainability fellowship.

REFERENCES


