Toward Understanding Life-Long Performance of a Sonexion File System

Mark Swan and Doug Petesch
Performance Team
Cray Inc.
St. Paul, MN, USA
dpetesch@cray.com

***Abstract*— Many of Cray's customers will be using their systems for several years to come. The one resource that is most affected by long-term use is storage. Files, both big and small, both striped and unstriped, are continually created and deleted, leaving behind free space of different sizes and in difference places on the spinning media. This paper will explore the effects of continual reuse of a Sonexion file system and a method of tuning the allocation parameters of the OSTs to minimize these effects.**

Keywords-Sonexion; I/O Performance

I. INTRODUCTION

When files are written to a Lustre Object Storage Target (OST), it is the ldiskfs layer that allocates space. In general, there are two things the allocator must know; where is the next free space and how much space should be allocated. To get a peek into the data structures used by the allocator, and actually affect some of its operations, we can look at /proc/fs/ldiskfs/<device>. In this directory, on a Sonexion, are three files; mb_groups, mb_last_group, and prealloc_table.

The mb_groups file is a representation of the free space on the OST. Each OST is divided into "multi-block allocation groups". By looking at the "tune2fs -l" output for an OST, one can see that there are 32,768 blocks per group and each block is 4,096 bytes. Therefore, each mb_group is 128 MiB of file system space. The mb_groups file can be displayed with "cat", but it is a big file.

The mb_last_group file indicates where the allocator will begin looking for free space. The value of this variable begins at zero when the Object Storage Server (OSS) is started. As data is written to the OST, this value continues incrementing until it reaches the end of the OST. At that point, the value wraps around to zero. While the allocator continues moving this pointer and allocating space, there are files being deleted whose data had been allocated in other parts of the OST. This constant allocation and deletion of data is what causes fragmentation. This file can be displayed with "cat" and can also be changed by using "echo" and redirection. By changing the value of this file, one can reposition where the allocator will begin searching for free space.

The prealloc_table file represents information about how much space will be allocated for each incoming stream of data to be written to the OST. This is commonly referred to as "OST pre-allocation". This file can be displayed with "cat" and can also be changed by using "echo" and redirection. The default values in this table for a Sonexion 1600 represents 1 MiB.

II. HYPOTHESIS

It is our belief that different values of OST pre-allocation have different effects on write and read rates as the OST becomes fragmented. We believe that a larger OST pre-allocation size will leave behind larger holes as files are deleted and, therefore, reduce the effects of fragmentation on bandwidth throughout the life of the file system.

We will be observing and manipulating the mb_groups, mb_last_group, and prealloc_table files in order to perform our experiments.

III. TESTING ENVIRONMENT

We will be using a Cray XE6 mainframe connected to a 4 SSU Sonexion 1600 with 3 TB drives. Each OST has about 23 TB (20.8 TiB) of available space. This system uses a 3:2 Fine Grained Routing (FGR) ratio. This means there are three LNET router connections servicing each LNET group and there are two OSSs in each LNET group for a total of four LNET groups.

The Lustre client is version 2.5.1 and tuned such that osc.*.max_rpcs_in_flight=64 and osc.*.max_dirty_mb=256. The Sonexion is running NEO 1.2.1 and Lustre server version 2.1.

IV. TESTING METHODOLOGY

The general method of our experiments is:

1) Decide how much space on each OST to use.

2) Choose a section of each OST within an SSU that is empty and near the fast zones.

3) Set 4 different SSUs to 4 different OST pre-allocation sizes (1, 2, 4, and 8 MiB).

4) Run optimal write and optimal read IOR jobs in each SSU to determine best rates. These will become our base rates for when OSTs are 100% free.

5) Simultaneously create 20 files on each OST so that data from all 20 files is intermingled.

6) Repeat until 100% free again:

a) Delete 1 file from each OST which releases 5% of the used space in our target areas.

b) Run optimal write and optimal read IOR jobs.

7) Record write and read rates, plot results, and compare to 100% free rates.

V. How much space to use

We will be creating 20 files on each OST. Given the nominal estimate of 5 GB/s/SSU, that is 625 MB/s per OST. We want to be able to write or read for 5 minutes even when our target area is highly full and fragmented. Writing 20 files for 5 minutes at 625 MB/s means we will be creating 187,500 MB of data or 1,464 multi-block allocation groups (128 MiB each). Considering that when we are 90% full, we need that much space to still be free, we should create 10 times that much data. Let's say 15,000 groups or 1,920,000 MiB or 1,875 GiB. Since we are creating 20 files, each file must be at least 93.75 GiB.

VI. Choosing a section of each OST

We will be testing the performance of entire SSUs on a file system that has been in production use for some time. Since each OST has a different amount of free space and in different places on the media, we analyzed each OST's allocation bitmap to find our target areas. Figures 1 is a depiction of the bitmap for OST 0000 before our testing began. The area beginning at group 60000 appears to be relatively empty and close enough to the beginning (fast edge) of the media that we will not be affected by slow-zone performance. As noted previously, we will be operating on a region that consumes 15,000 groups. Figure 2 is a depiction of the bitmap of OST 0000 after we wrote 20 94-GiB files at location 60000.

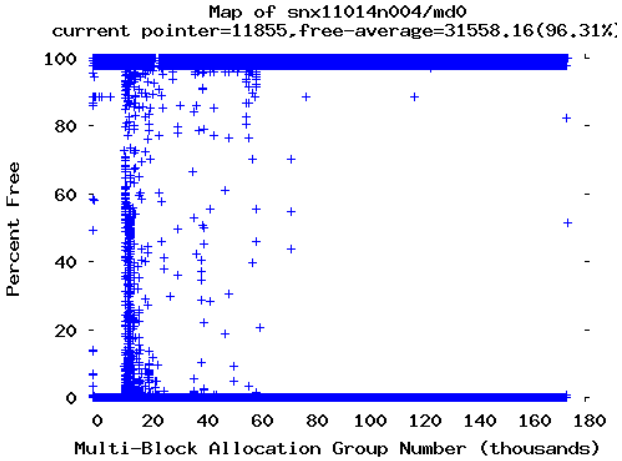

Figure 1, OST0000 before preserving space

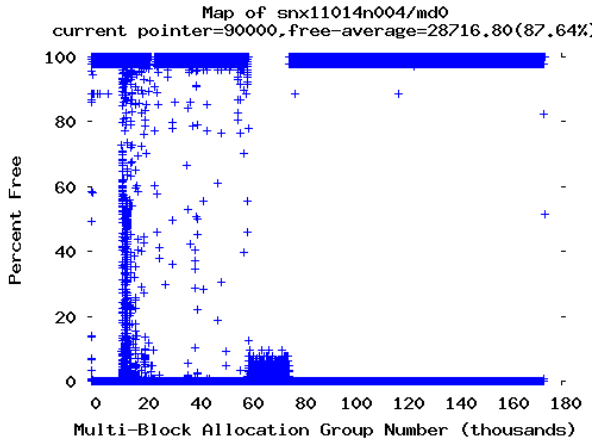

Figure 2, OST0000 after preserving space.

VII. Analyzing Space Allocation

Using the filefrag utility, we can examine the physical characteristics of each of the 20 files that we created on our OSTs. The following plots show the distribution of contiguous space for all of the OSTs that have 1, 2, 4, and 8 MiB OST pre-allocation sizes. As we can see, the contiguous space for each setting is dominated by a size equal to the setting. This shows that; a) the ldiskfs allocator used the pre-allocation size we specified and b) the 20 files were highly intermingled. If the files were not highly intermingled, there would be many more areas of contiguousness that are larger than the specified pre-allocation size.

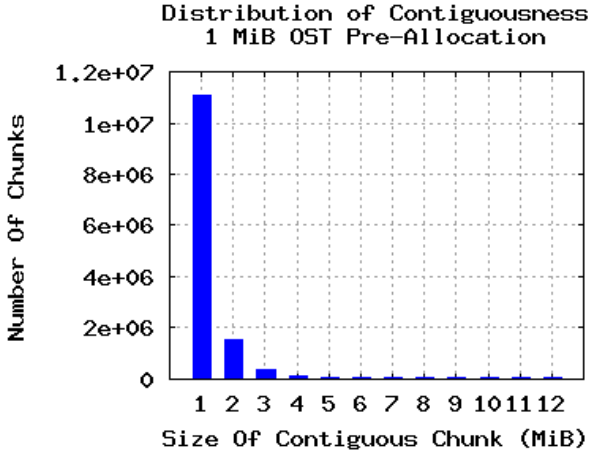

Figure 1, Distribution of Contiguousness for 1 MiB OST Pre-allocation

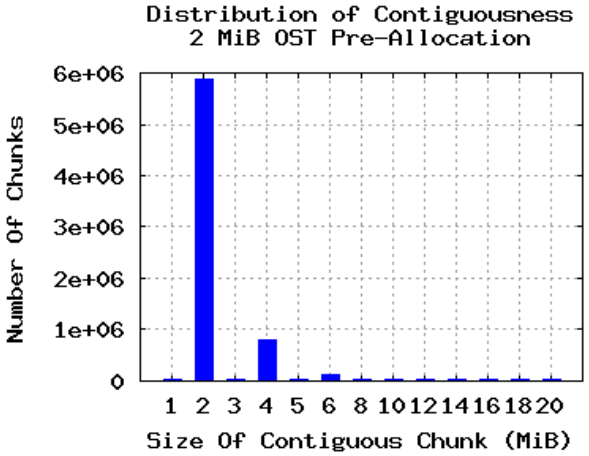

Figure 2, Distribution of Contiguousness for 2 MiB OST Pre-allocation

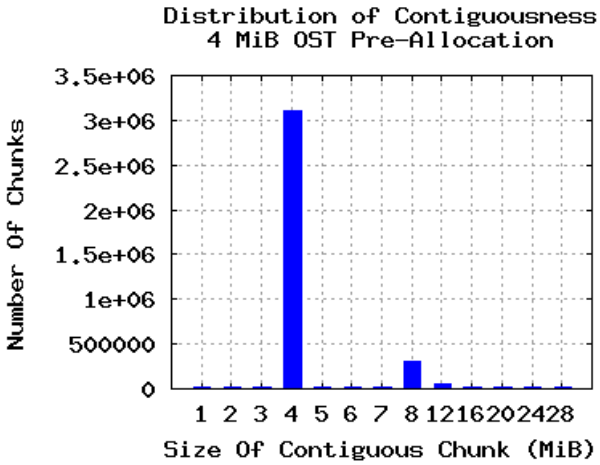

Figure 3, Distribution of Contiguousness for 4 MiB OST Pre-allocation

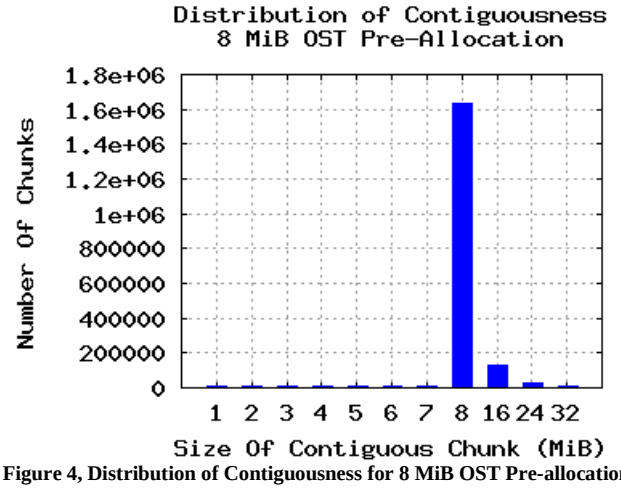

Figure 4, Distribution of Contiguousness for 8 MiB OST Pre-allocation

VIII. RESULTS

Figures 7, 8, 9 and 10 show the per-SSU write and read rates for each of the OST pre-allocation sizes.

As expected, using OST pre-allocation sizes greater than 1 MiB cause the write rates to decrease due to seeking [1]. At 1 MiB, both write and read rates fall dramatically as soon as there is as little as 5% fragmentation. Larger OST pre-allocation sizes result in larger contiguous holes left behind due to file removal and so future I/O rates show very little decrease from the base.

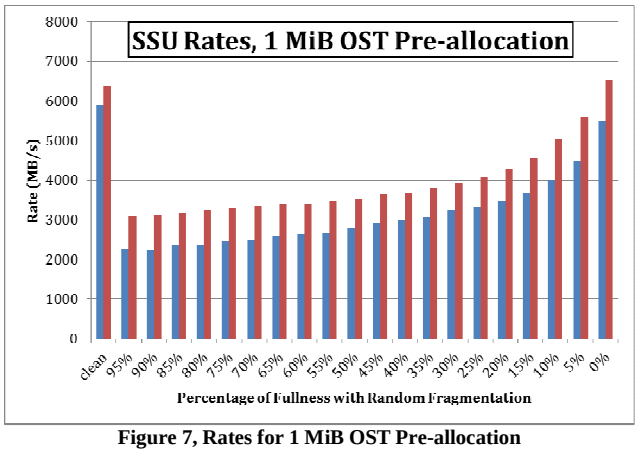

Figure 7, Rates for 1 MiB OST Pre-allocation

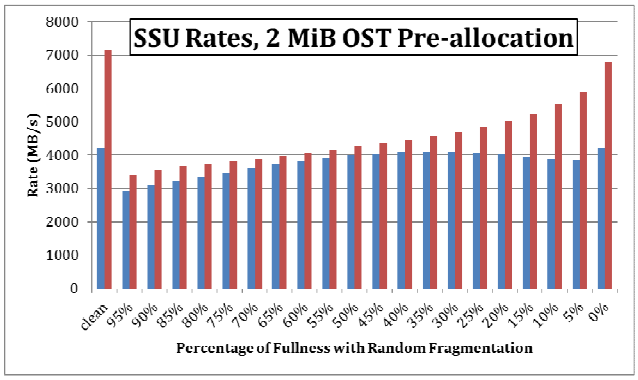

Figure 8, Rates for 2 MiB OST Pre-allocation

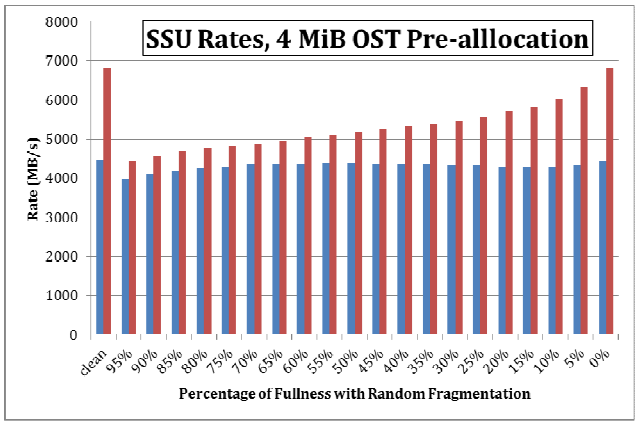

Figure9, Rates for 4 MiB OST Pre-allocation

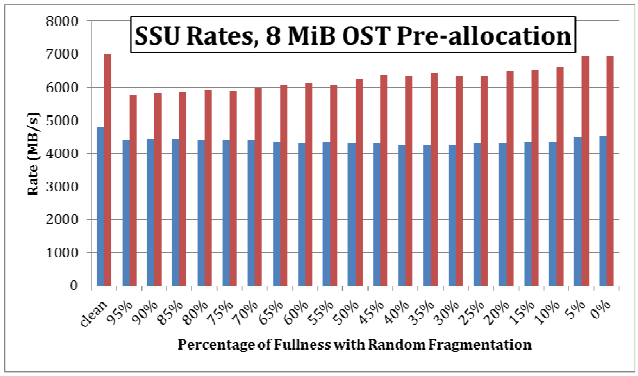

Figure 10, Rates for 8 MiB OST Pre-allocation

IX. SUMMARY

At different levels of fragmentation, the worst value of OST pre-allocation is 1 MiB. This is the value set at the factory for Sonexion 1600. Cray has not been suggesting changes to this value.

The best value of OST pre-allocation is 8 MiB. Using 8 MiB OST pre-allocation, no matter what level of fragmentation, the following observations can be made:

A. Optimal write rate is at least 89% of a clean file system.

B. Optimal read rate is at least 82% of a clean file system.

C. Optimal write rates are always above 4 GB/s/SSU.

D. Optimal read rates are always above 5 GB/s/SSU.

X. FUTURE AREAS OF RESEARCH

All the side effects of large OST pre-allocation sizes are not yet known. We would like to investigate how space is allocated for files that are striped across many OSTs. We would like to understand how the ldiskfs allocator fills in small holes when the pre-allocation size is large. We would like to understand if the use of 4M RPCs affects the outcome of this research.

This study should also be performed on the Sonexion 2000 product where the default pre-allocation is 16 MiB.

REFERENCES

[1] M. Swan, “Tuning and Analyzing Sonexion Performance” CUG 2014.