

Towards Smoothing Data Movement Between RAM and Storage

Extended Abstract

Tariq Alturkestani

King Abdullah University of Science and Technology

Thuwal, Saudi Arabia

tariq.alturkestani@kaust.edu.sa

Vincent Etienne

Saudi Aramco

Dhahran, Saudi Arabia

vincent.etienne@aramco.com

Thierry Tonellot

Saudi Aramco

Dhahran, Saudi Arabia

thierrylaurent.tonellot@aramco.com

Hatem Ltaief

King Abdullah University of Science and Technology

Thuwal, Saudi Arabia

hatem.ltaief@kaust.edu.sa

ABSTRACT

We propose to design and implement a software framework, which provides a Multilayer Buffer System (MBS) to cache in/out datasets into CPU main memory from/to slower storage media, such as parallel file systems (e.g., Lustre), solid-state drive (e.g., Burst Buffer) or non-volatile RAM. Although MBS scope may be broad in terms of scientific applications, we focus on the RTM application as a proxy for I/O intensive workloads, since reading and writing are ubiquitous operations during the dumping phase of the source wavefield (forward propagation) as well as its retrieval phase (backward propagation) for the image condition calculations.

ACM Reference Format:

Tariq Alturkestani, Thierry Tonellot, Vincent Etienne, and Hatem Ltaief. 2018. Towards Smoothing Data Movement Between RAM and Storage : Extended Abstract. In *Proceedings of ACM Supercomputing conference (SC'18)*. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/nnnnnnn>. nnnnnnn

1 INTRODUCTION

Due to Moore's law, the performance of CPUs has been historically doubling every two years. However, storage systems have not been able to catch up in the same speed. I/O is considered a bottleneck for many critical simulations that rely on writing and reading intermediate results or snapshots during their computation. Reverse Time Migration (RTM)[1], for example, is a seismic imaging method that compute high resolution image from seismic data, by propagating waves in 3D models. Based on an adjoint-state formulation, RTM requires combining at regular time steps a forward-propagated source wavefield with a back-propagated receiver wavefield. This process involves thus a first phase where the 3D simulation states, or snapshots, of the source wavefield are computed and saved at predetermined time steps. Then in a second phase, the receiver wavefield is computed and the imaging time

steps, the source snapshots are retrieved and correlated with the receiver snapshots to update the final image. The time that is spent blocking for I/O during the two phases amount for about %68 of the entire execution time. Recent super computer architectures include several layers of storage from DRAM to BurstBuffer to disk, which triggered the development of libraries implementing the checkpoint restart pattern, mainly for fault tolerance purpose[2]. Utilization of these libraries for RTM, can improve the overall performances, mainly through asynchronous I/O accesses, which allow for a partial overlap with computations. However, most of these generic components are not optimal as they cannot benefit from the very regular access pattern in the RTM which allow for further performance gain through customized prefetching strategies. The system, Multilayer Buffer System (MBS), proposed in this research, shows a general and versatile method for overlapping I/O with computation that helps to reduce the blocking time through asynchronous access and an RTM specific stage out and prefetching strategy, while minimizing the impact on the computational kernel. MBS shows that coupling both storage and memory systems and using the knowledge of data access patterns can decrease the time spent blocking for I/O by more than %70 and thereby improving the entire execution time by 2.98X on regular storage and up to 3.95X on BurstBuffer.

2 EXPERIMENTATION

We began our journey of finding the best ways for overlapping I/O and computation with I/O profiling of the RTM application on Lustre and DataWarp file systems. The RTM migration implementation and parameters are based on Bausal et. al [1] and table 2. As an experimentation platform we used a super computer with the nodes specifications listed in table 3.

3 SUMMARY

The current I/O profiling reveals that more than two-third of the RTM total elapsed time is spent on I/O operations, when using a 530x530x530 grid size, 3500 time steps and I/O access frequency set to three, on a two-socket 16-core Intel Haswell system. By deploying customized prefetching strategies at the core of MBS, our library solution permits to significantly hide the overhead of I/O operations by issuing asynchronous I/O accesses. Indeed, MBS enables to create overlapping situations, in which data movement from/to slower storage media occurs while carrying on the stencil

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SC'18, November 2018, Dallas, Texas USA

© 2018 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn>

Medium	Wall (s)	Kernel (s)	%	Write (s)	%	Read (s)	%	misc	%
Lustre (1 OST)	4635.60	1091.96	23.56%	2380.03	51.34%	1137.15	24.53%	26.45	0.57%
Lustre (2 OST)	10836.80	1100.87	10.16%	1493.99	13.79%	8213.33	75.79%	28.61	0.26%
Lustre (4 OST)	8414.49	1093.58	13.00%	2024.14	24.06%	5220.52	62.04%	76.24	0.91%
Lustre (6 OST)	8232.92	1100.82	13.37%	2040.64	24.79%	5060.68	61.47%	30.78	0.37%
Lustre (8 OST)	7966.05	1101.11	13.82%	1993.10	25.02%	4831.36	60.65%	40.48	0.51%
BurstBuffer	1833.80	1091.67	59.53%	400.09	21.82%	312.44	17.04%	29.59	1.61%

Table 1: Execution time breakdown of RTM using Lustre and BurstBuffer as a storage medium. On Lustre we varied the number of Object Storage Target (OSTs) to find the best one

Grid Size	$(512 + 18)^3$
Kernel OMP threads	567.92 MB
Snap Size	$(512 + 18)^3$
Number of time steps	3500
Snap IO frequency	3
Number of IO snaps	1167
Total snaps size	647.78 GB
Total IO traffic	1.26 TB

Table 2: Experimentation setup of RTM

Number of nodes	6174 (Two processors per node)
Processor	Intel Haswell E5-2698 V3 (16 cores)
Memory	128 GB
Memory Bandwidth	136 GB/s

Table 3: Experimentation platform node configuration

MBS Helpers	MBS RAM	Wall	Kernel	Write	Read	Overall IO
1	1	3.95	0.98	94.01	11,137.64	138.37
	16	3.92	0.98	94.93	75,308.17	140.21
	64	3.77	0.98	95.19	77,887.21	140.59
2	1	3.65	0.90	95.10	20,600.60	140.23
	16	3.16	0.78	94.99	71,072.08	140.29
	64	3.09	0.79	95.21	70,194.65	140.60
4	1	3.65	0.90	95.07	26,819.65	140.25
	16	2.93	0.72	95.45	68,918.38	140.97
	64	2.90	0.74	95.46	77,357.37	140.98
8	1	3.64	0.90	95.18	16,552.45	140.26
	16	2.93	0.72	95.08	70,194.65	140.42
	64	2.89	0.74	95.52	72,430.15	141.07
16	1	3.54	0.87	96.27	26,819.65	142.02
	16	2.79	0.69	95.76	66,113.56	141.42
	64	2.77	0.70	96.12	73,364.73	141.95

Table 5: Speed up breakdown on BurstBuffer after attaching MBS helpers. 1 OST of Lustre is used as a base.

MBS Helpers	MBS RAM	Wall	Kernel	Write	Read	Overall IO
1	1	1.50	0.97	2.84	1.05	
	16	1.47	0.97	2.95	0.95	
	64	1.71	0.97	3.34	1.47	
2	1	1.50	0.89	3.03	1.07	
	16	2.87	0.83	18.2	9.23	
	64	2.98	0.83	33.03	14.54	
4	1	1.41	0.89	2.89	0.94	
	16	2.97	0.73	92.36	5,058.51	
	64	2.94	0.76	92.71	81.04	
8	1	1.42	0.89	2.77	0.98	
	16	2.89	0.71	91.4	68,503.21	
	64	2.85	0.73	92.47	67,687.70	
16	1	1.48	0.87	2.87	1.10	
	16	2.77	0.68	92.83	47,579.64	
	64	2.76	0.70	93.21	59,226.73	

Table 4: Speed up breakdown on Lustre after attaching MBS helpers. 1 OST of Lustre is used as a base.

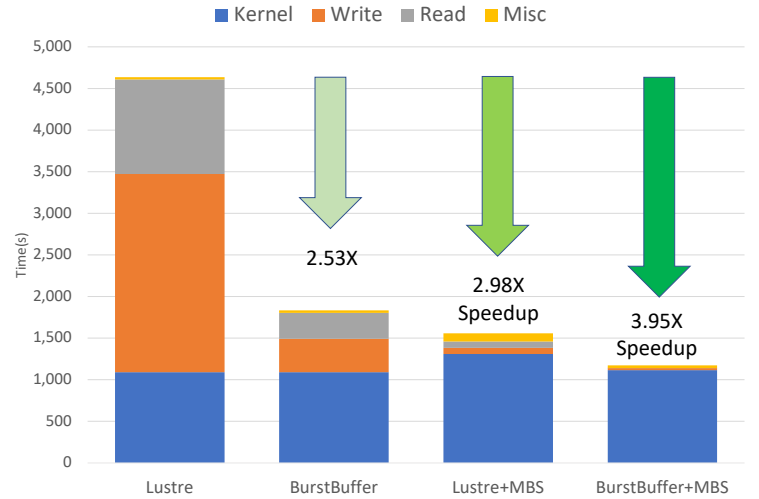


Figure 1: Execution time breakdown. We can clearly see how MBS helpers help to overlap compute and IO.

computations. The application turns out to perform at the speed as if it runs in CPU-memory resident mode of operation.

REFERENCES

- [1] Edip Baysal, Dan D Kosloff, and John WC Sherwood. 1983. Reverse time migration. *Geophysics* 48, 11 (1983), 1514–1524.
- [2] P.W.D. Charles. 2013. Project Title. <https://github.com/charlespwd/project-title>.