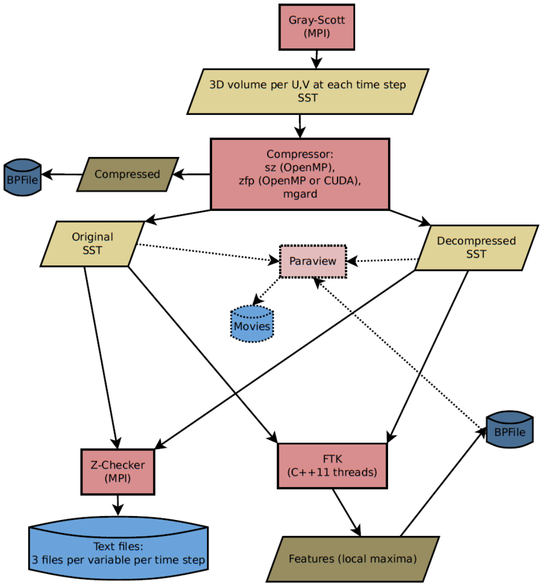
Compression pipeline

Igor Yakushin

October 23, 2019

Pipeline


Gray-Scott
(MPI)
3D volume per U,V at each time step
SST
Compressor:
sz (OpenMP),
zfp (OpenMP or CUDA),
mgard
Compressed
BPFile
Original
SST
Paraview
Decompressed
SST
Movies
BPFile
Z-Checker
(MPI)
FTK
(C++11 threads)
Text files:
3 files per variable per time step
Features (local maxima)

Pipeline

- Gray-Scott simulation
  - currently uses 256^3 grid
  - two variables, U and V, evolve according to two coupled non-linear PDEs, generating non-trivial patterns
  - initial condition: U is 1 everywhere except for the cube in the middle where it is 0.25; V is 0 everywhere except for the same cube in the middle where it is 0.33
  - some random noise is added to U at each iteration, its amplitude is a parameter
  - 1000 time steps
  - every 100 steps a checkpoint is created and sent via ADIOS2 SST stream to compressor

Pipeline

- Compressor
  - uses one of the following algorithms: SZ, ZFP, MGARD
  - compresses and decompresses the data
  - compressed data is written to a file with enough metadata to decompress
  - original and lossy (compressed/decompressed) data are sent via ADIOS2 SST streams to ZChecker and FTK to evaluate the quality of the compression
- ZChecker
  - gets original and lossy data
  - runs its usual comparison and stores the results in usual ZChecker format: 3 text files per iteration, per variable

Pipeline

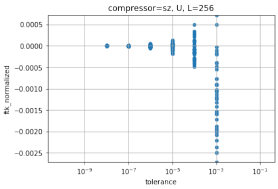
- FTK
  - gets original and lossy data
  - finds the number of local maxima on each variable on both datasets
  - computes the distance as normalized difference between the number of local maxima for each variable
  - local maxima coordinates and values, number of maxima, distances are stored in ADIOS2 BP4 files.
- Cheetah/Savanna is used to set up and run the experiments
- ParaView
  - used for debugging purposes offline
  - compared original and lossy data side by side to visualize time evolution and spacial distribution of U and V and show local maxima found by FTK
  - see the link to movies on 32^3 grid in references

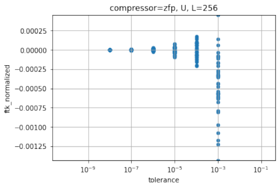
FTK

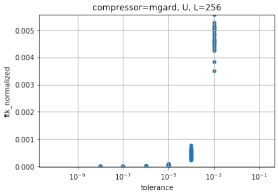
- We tried to use FTK as a feature detector to compare the original and lossy data
- As a distance, we use normalized difference of the number of local maxima
- Next slide shows how this distance depends on the tolerance that is a parameter for each compressor
- ZFP and MGARD have only one input parameter to control the compression
- SZ can use several kinds of tolerance parameters, here by “tolerance” we mean the default one: `pw_relBoundRatio`

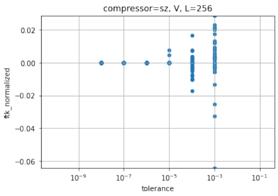
FTK: distance

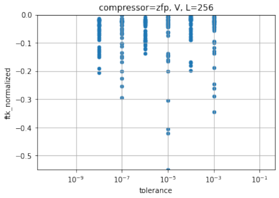
There is no clear correlation between tolerance and FTK distance

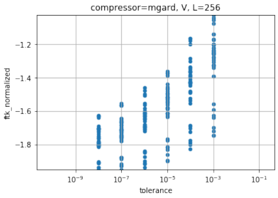






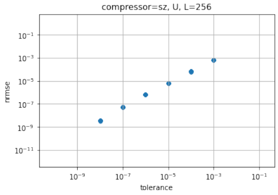


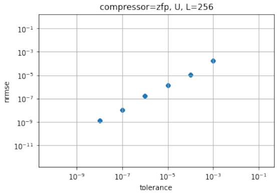


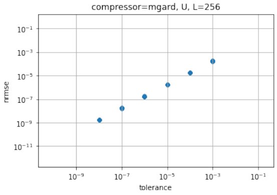


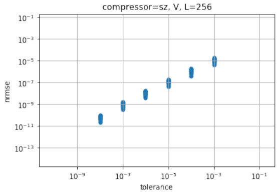
FTK: ZChecker measurements

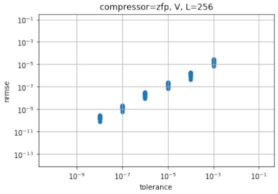
Varios ZChecker's measurements behave as expected

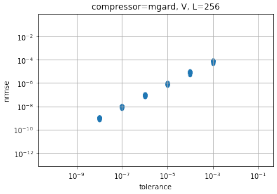












FTK: number of maxima for *noise* $= 10^{-2}$, SZ

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	828279	828279	0.0	0.00e+00
1.0e-07	825165	825166	-1.0	-1.21e-06
1.0e-06	825966	825996	-30.0	-3.63e-05
1.0e-05	824939	825007	-68.0	-8.24e-05
1.0e-04	827345	827418	-73.0	-8.82e-05
1.0e-03	824677	825416	-739.0	-8.96e-04

Table: SZ, U, *step* $= 1000$, *noise* $= 10^{-2}$

Notice: the number of maxima in the original data above is only 20 times smaller than the total number of points in the volume.

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	981	981	0.0	0.00e+00
1.0e-07	1004	1004	0.0	0.00e+00
1.0e-06	917	917	0.0	0.00e+00
1.0e-05	1003	1003	0.0	0.00e+00
1.0e-04	987	985	2.0	2.03e-03
1.0e-03	941	942	-1.0	-1.06e-03

Table: SZ, V, *step* $= 1000$, *noise* $= 10^{-2}$

FTK: number of maxima for *noise* $= 10^{-2}$, SZ

- For V the number of maxima is of reasonable size and almost identical between the original and lossy data; a small difference appears only at high tolerance 10^{-4}
- The difference between the graphs and the tables is coming from two sources:
  - In the tables only the last checkpoint is shown while on the graph all 10 checkpoints
  - In the tables the results shown only for 64 MPI ranks for Gray-Scott and ZChecker (this experiment was done on theta) while on the graphs 4 combinations of $(32, 64)$ ranks are tried

FTK: number of maxima for $noise = 10^{-2}$, ZFP

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	825854	825854	0.0	0.00e+00
1.0e-07	827774	827777	-3.0	-3.62e-06
1.0e-06	826774	826766	8.0	9.68e-06
1.0e-05	826603	826629	-26.0	-3.15e-05
1.0e-04	825891	825910	-19.0	-2.30e-05
1.0e-03	826406	826549	-143.0	-1.73e-04

Table: ZFP, U, $step = 1000$, $noise = 10^{-2}$

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	914	933	-19.0	-2.06e-02
1.0e-07	1064	1092	-28.0	-2.60e-02
1.0e-06	1018	1117	-99.0	-9.29e-02
1.0e-05	987	994	-7.0	-7.08e-03
1.0e-04	964	978	-14.0	-1.44e-02
1.0e-03	975	997	-22.0	-2.23e-02

Table: ZFP, V, $step = 1000$, $noise = 10^{-2}$

While the results for U are similar to those of SZ, the results for V show no clear correlation between the number of maxima and the tolerance

FTK: number of maxima for *noise* $= 10^{-2}$, MGARD

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	826506	826502	4.0	4.84e-06
1.0e-07	825201	825204	-3.0	-3.64e-06
1.0e-06	826299	826302	-3.0	-3.63e-06
1.0e-05	827031	827031	0.0	0.00e+00
1.0e-04	824480	824156	324.0	3.93e-04
1.0e-03	825847	821624	4223.0	5.13e-03

Table: MGARD, U, $step = 1000$, $noise = 10^{-2}$

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	979	9540	-8561.0	-1.63e+00
1.0e-07	955	7887	-6932.0	-1.57e+00
1.0e-06	927	6472	-5545.0	-1.50e+00
1.0e-05	900	4816	-3916.0	-1.37e+00
1.0e-04	934	3720	-2786.0	-1.20e+00
1.0e-03	966	3058	-2092.0	-1.04e+00

Table: MGARD, V, $step = 1000$, $noise = 10^{-2}$

Notice: for V MGARD lossy data has 10 times more local maxima than the original data and as the tolerance increases, the distance decreases which is opposite to what we saw for other compression algorithms. Apparently MGARD introduces its own artifacts and the higher the required compression precision, the more artifacts are introduced.

FTK: number of maxima for *noise* $= 0$

- Reducing the noise amplitude from 10^{-2} to 10^{-5} does not significantly change the results: still the number of local maxima in the original data for U is only about 20 times less than the number of points in the volume - anything, no matter how small, that is larger than its immediate neighbors is considered to be local maxima
- Therefore, let us try to turn the noise completely off
- We shall consider only two values for tolerance for simplicity

FTK: number of maxima for *noise* = 0

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	50	645	-595.0	-1.72e+00
1.0e-03	50	452	-402.0	-1.61e+00

Table: SZ, U, step=500, $noise = 0$

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	558	614	-56.0	-9.57e-02
1.0e-03	558	551	7.0	1.27e-02

Table: SZ, V, step=500, $noise = 0$

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	50	219	-169.0	-1.27e+00
1.0e-03	50	235	-185.0	-1.31e+00

Table: ZFP, U, step=500, $noise = 0$

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	558	579	-21.0	-3.70e-02
1.0e-03	558	617	-59.0	-1.01e-01

Table: ZFP, V, step=500, $noise = 0$

FTK: number of maxima for *noise* $= 0$

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	50	4687	-4637.0	-1.96e+00
1.0e-03	50	1224	-1174.0	-1.85e+00

Table: MGARD, U, step=500, *noise* $= 0$

tolerance	number of max, original	number of max, lossy	difference	normalized difference
1.0e-08	558	6480	-5922.0	-1.68e+00
1.0e-03	558	1860	-1302.0	-1.08e+00

Table: MGARD, V, step=500, *noise* $= 0$

- While setting *noise* $= 0$ got rid of unmanageable number of local maxima in the original data, it did not address compression/decompression artifacts generated in the lossy data
- Hanqi suggested to try to use Gaussian filtering to precondition data before applying FTK - almost finished implementing it

Offloading Gray-Scott to GPU with Kokkos

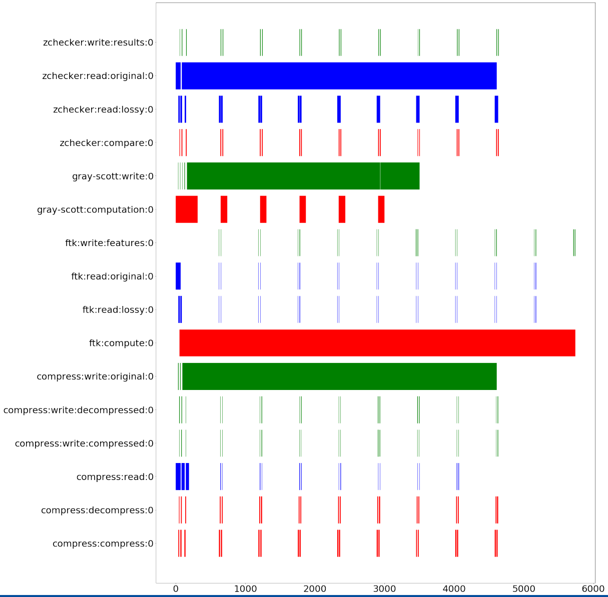
- The original version of Gray-Scott runs on CPU using MPI
- To justify using Summit, Gray-Scott was accelerated with Kokkos
- The same code - one line change - can be used to offload to GPU on Summit or to use multithreading on Theta

	CPU	CPU	CPU	CPU	GPU	GPU	GPU
MPI ranks	1	8	21	42	1	3	6
compute/rank/iteration, ms	199,972	23,743	9,911	4968	1,819	625	404
write/rank/iteration, ms	7,597	3,414	531	335	7850	2887	1631

Table: L=512, summit

- To use Kokkos requires some bleeding edge compiler specifically built to offload to GPU, same applies to OpenMP GPU offloading
- I am using LLVM 9.0.0 on Summit
- On Theta, while I can build Kokkos version of Gray-Scott, I had some trouble compiling some other components of the pipeline. So for now I am using original version of Gray-Scott on Theta.

Gantt chart on Theta



Gantt chart on Theta

- Timing is shown for SZ, U, 256^3 grid
- Only one MPI rank of each pipeline component is shown, other ranks have identical timing
- Red corresponds to computations: Gray-Scott compute, compression/decompression, ZChecker and FTK comparisons
- Green corresponds to write
- Blue corresponds to read
- Everything is dominated by FTK computations, all other computations take negligible time in comparison
- Long times for I/O are misleading. ADIOS2 SST streams are configured in such a way that
    - Gray-Scott waits for compressor to pick up its checkpoint before proceeding to the next iteration
    - compressor waits for ZChecker and FTK to pick up its data before proceeding to the next iteration
- As a result, long I/O times are due to waiting for FTK to finish the previous iteration and not due to actually moving bits

Gantt chart on Theta: adios2.xml

```
<adios-config>
 <io name="SimulationOutput">
  <engine type="SST">
  <parameter key="RendezvousReaderCount" value="1"/>
  <parameter key="QueueLimit" value="1"/>
  <parameter key="QueueFullPolicy" value="Block"/>
 </engine>
</io>
<io name="OriginalOutput">
 <engine type="SST">
  <parameter key="RendezvousReaderCount" value="2"/>
  <parameter key="QueueLimit" value="1"/>
  <parameter key="QueueFullPolicy" value="Block"/>
 </engine>
</io>
<io name="CompressedOutput">
 <engine type="BPFile">
  <parameter key="RendezvousReaderCount" value="1"/>
  <parameter key="QueueLimit" value="1"/>
  <parameter key="QueueFullPolicy" value="Block"/>
 </engine>
</io>
```

```
<io name="DecompressedOutput">
 <engine type="SST">
  <parameter key="RendezvousReaderCount" value="2"/>
  <parameter key="QueueLimit" value="1"/>
  <parameter key="QueueFullPolicy" value="Block"/>
</engine>
</io>
<io name="FTK">
 <engine type="BPFile">
  <parameter key="RendezvousReaderCount" value="1"/>
  <parameter key="QueueLimit" value="1"/>
  <parameter key="QueueFullPolicy" value="Block"/>
 </engine>
</io>
</adios-config>
```

Detailed comparison

	Summit			Theta	
	MPI ranks	CPU threads	GPUs	MPI ranks	CPU threads
Gray-Scott	2	1	2	64	1
SZ	1	1	0	1	1
ZFP	1	1	0	1	1
MGARD	1	1	0	1	1
ZChecker	5	1	0	64	1
FTK	1	20	0	1	64

Table: Utilized computational resources

	Summit	Theta
Gray-Scott compute	14 ± 0.05	32 ± 0.2
SZ compress	2.5 ± 0.05	11 ± 0.5
SZ decompress	2.0 ± 0.15	7.9 ± 0.5
ZFP compress	0.4 ± 0.3	1.3 ± 1.1
ZFP decompress	0.43 ± 0.35	1.5 ± 1.2
MGARD compress	15 ± 6	72 ± 48
MGARD decompress	2.2 ± 0.07	16 ± 0.7
ZChecker compare	1.9 ± 0.009	3.5 ± 0.1
FTK compare	81 ± 0.3	560 ± 0.8

Table: Average timing of various pipeline components in seconds. Grid size 256^3. Gray-Scott simulation was run 1000 time steps, saving checkpoints every 100 iterations. Therefore the averaging was done over 10 samples. The input compression tolerance here is fixed at 10^{-3}.

Detailed comparison

- On Summit the whole pipeline fits into a single node
- On Theta each component runs on a separate node - 4 nodes
- Notice that timing for SZ compression/decompression is measured differently than for ZFP and MGARD: SZ times include both U and V variables, while for the other compression algorithms time is measured separately for different variables and this is the origin of high std for ZFP and MGARD and higher mean for SZ. This needs to be unified later.
- If instead of 64 hardware threads, one uses all 256 hardware threads on theta's KNL node, FTK compare time drops from 560 to 480 ± 0.6 seconds.
- If instead of 64 MPI ranks for Gray-Scott on Theta, one uses 32 MPI ranks, the run time for 100 iterations increases from 32 to 62 ± 0.3 seconds.
- If instead of 64 MPI ranks for ZChecker comparison on theta, one uses 32 MPI ranks, the run time increases from 1.9 to 7 ± 0.2 seconds.

Detailed comparison

- FTK is the biggest bottleneck
- MGARD is much slower than other compression algorithms
- Theta is much slower than Summit
- Possible improvements to timing:
    - FTK is called 4 times sequentially for original and lossy data, for U and V - this can be done in parallel using MPI; debugging the corresponding version
    - Currently FTK can only use CPU threads - accelerate it to use GPUs
    - Hanqi's summer students integrated FTK with MPI - might help on Theta but not on Summit
    - Recent development in MGARD - it is cudarized now

TODO

- Try to reduce the influence of noise in the original data and compression/decompression artifacts in the lossy data on FTK distance by using something like Gaussian filtering
- If the above helps, speed up FTK
  - Use MPI to call FTK in parallel 4 times
  - Accelerate FTK on GPU?
  - Use MPI version of FTK once Hanqi releases it
- Try newer accelerated version of MGARD
- Benchmark Kokkos and original version of Gray-Scott on Theta
- Unify how compression/decompression timing is done for all the compression algorithms
- Visualize 256^3 with ParaView after all the changes since the previous set of movies was generated
- Resolve LLVM compiling issues on Theta
- Debug OpenMP GPU offloading version of Gray-Scott?
- Put the pipeline into Spack? Singularity container?

References

Pipeline repo:
https://github.com/CODARcode/cheetah/tree/compression_split/examples/05-gray-scott-compression
Current work page:
https://users.nccs.gov/~iyakushin/compression_pipeline/index.html
Paraview movies:
https://anl.app.box.com/s/c7x29xf49mofmeftwqrlqe4nt6heulyh

K. Mehta, B. Allen, M. Wolf, J. Logan, E. Suchyata, J. Choi, K. Takahashi, I. Yakushin, T. Munson, I. Foster, and S. Klasky
A codesign framework for online data analysis and reduction at the exascale. 2019.
Accepted for presentation at WORKS 2019

The "Parallel Vectors" Operator - A Vector Field Visualization Primitive
Ronald Peikert and Martin Roth

Cheetah/Savanna repo: https://github.com/CODARcode/cheetah

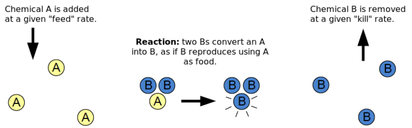
ADIOS2: https://adios2.readthedocs.io

Tao, D., Di, S., Guo, H., Chen, Z., & Cappello, F. (2019).
Z-checker: A framework for assessing lossy compression of scientific data.
The International Journal of High Performance Computing Applications, 33(2), 285-303.

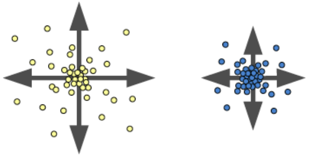
Z-checker repo: https://github.com/CODARcode/Z-checker
SZ repo: https://github.com/disheng222/SZ
MGARD repo: https://github.com/CODARcode/MGARD
ZFP repo: https://github.com/LLNL/zfp
FTK repo: https://github.com/CODARcode/ftk

Gray-Scott: https://www.karlsims.com/rd.html

Apendix A: Gray-Scott reaction-diffusion model


Chemical A is added at a given "feed" rate.
A
A
A
Reaction: two Bs convert an A into B, as if B reproduces using A as food.
B B
A
B B
B
Chemical B is removed at a given "kill" rate.
B
B
B

Diffusion: both chemicals diffuse so uneven concentrations spread out across the grid, but A diffuses faster than B.



The grid is repeatedly updated using the following equations to update the concentrations of A and B in each cell, and model the behaviors described above.

$$A' = A + (D_A \nabla^2 A - AB^2 + f(1-A))\Delta t$$
$$B' = B + (D_B \nabla^2 B + AB^2 - (k+f)B)\Delta t$$

New values

Previous values

Diffusion: rates for A and B

These are 2D Laplacian functions, which give the difference between the average of nearby grid cells and this cell. This simulates diffusion because A and B become more like their neighbors.

Feed: at rate *f*, scaled by (1-A) so A doesn't exceed 1.0

"Delta t" is the change in time for each iteration. All the terms get scaled by this.

Kill: this term is subtracted to remove B and scaled by B so it doesn't go below 0. *f* is added to *k* here so the resulting kill rate is never less than the feed rate.

Reaction: the chance that one A and two B will come together is *A x B x B*. A is converted to B so this amount is subtracted from A and added to B.