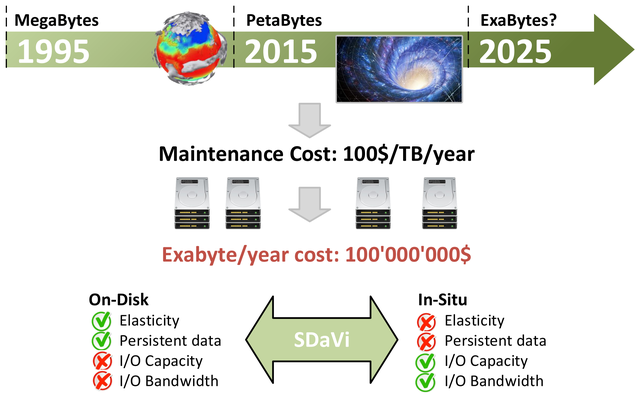
Virtualized Big Data: Reproducing Simulation Output on Demand

Salvatore Di Girolamo, Torsten Hoefler (advisor) - *ETH Zurich*

Teaser & Motivation



Abstract

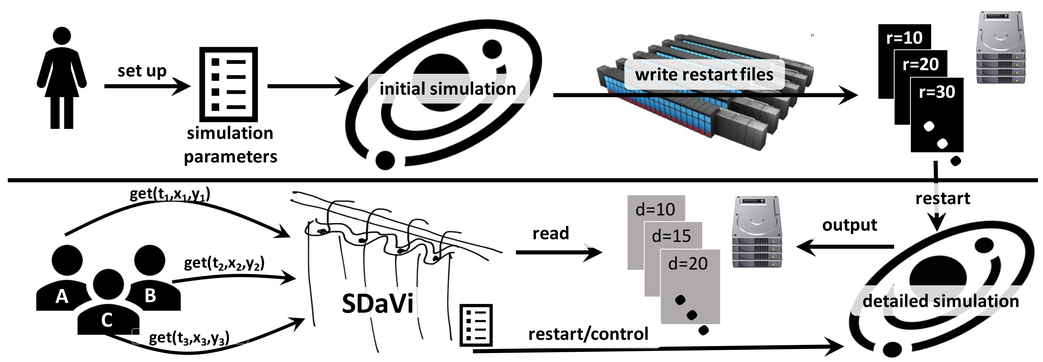
Scientific simulations are being pushed to the extreme in terms of size and complexity of the addressed problems, producing astonishing amount of data.

If the data is stored on disk, analysis applications can randomly access simulation output. Yet, storing the massive amounts of ever-growing simulation output data is a large part of the big data challenge in scientific computing. This is primarily due to the high storage costs and the fact that compute capabilities grow faster than storage capacities and bandwidths. In situ analysis removes the storage costs by processing the data as it is generated but applications lose random access.

We propose to not store the full simulation output data but to produce it on demand. Our system – SDaVi, Simulation Data Virtualizer - intercepts I/O requests of both analysis tools and simulators, enabling data virtualization. This new paradigm allows us to explore the computation-storage tradeoff, by trading computation power for storage space. Hence, SDaVi can shift towards the on-disk or in-situ approach according to the available/assigned storage space.

We apply SDaVi to applications employing typical data traversal patterns in two different contexts: climate modeling (COSMO) and multiphysics simulations (FLASH). SDaVi offers a viable path towards exa-scale scientific simulations, by exploiting the growing computing power and relaxing the storage capacity requirements.

SDaVi: Simulation Data Virtualizer



Interfacing the Analysis Tools

Virtualization-Oblivious:
Analysis tool is not aware of the virtualized enviroment. Requests are expressed as calls to common I/O libraries and served one-by-one.

Operation	(P)NetCDF	(P)HDF5	ADIOS
open	nc(mpi)_open	H5Fopen	adios_open
put	nc(mpi)_vara_put_type	H5Dwrite	adios_write
get	nc(mpi)_vara_put_type	H5Dread	adios_schedule_read
close	nc(mpi)_close	H5Fclose	adios_close

Virtualization-Aware:
Additional hints on the access pattern are provided.


```
int sdavi_request_range(char * begin, char * end, int stride)
```

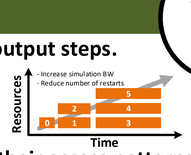
Synchronized:
Sets of analysis tools operate in a block-synchronous manner.

(P)netCDF / HDF-5 DV Client Library

Control (TCP)

Data Virtualizer

Restarts simulations for missing output steps.
How to minimize computing resources?
Horizontal Prefetching: longer simulations for constant-trajectory analysis tools



Keeps track of analysis tools and their access patterns.
How to optimize the response time?
Vertical Prefetching: Increase the number of parallel re-simulation to match analysis tool's BW

Manages the cache hierarchy.
What is the best caching strategy in this context?

Control (TCP)

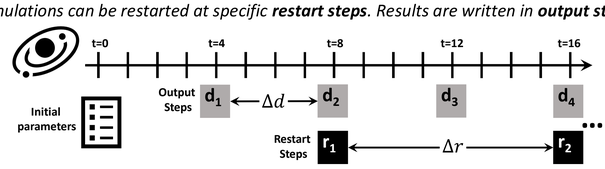
DV Simulator Library (P)netCDF / HDF-5

DATA

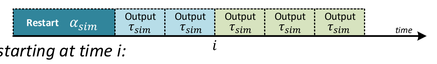
DATA

On-Demand Simulations

Modelling Simulations
*Simulations can be restarted at specific **restart steps**. Results are written in **output steps**.*



Performance Model
Time to simulate n output steps starting at time i:



$$T(i,n) = \alpha_{sim} + (i \bmod \Delta r) \cdot \tau_{sim} + n \cdot \tau_{sim}$$

SDaVI profiles simulation instances and updates the models' parameters at runtime.

How to interface a simulator?
A set of simulator-specific functions (i.e., convert filenames in integer keys) can be provided as a LUA configuration file.

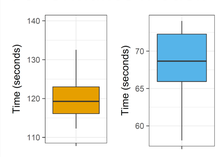
Experiments

Performance Model calibrated on:
Piz Daint @ CSCS: Cray XC40

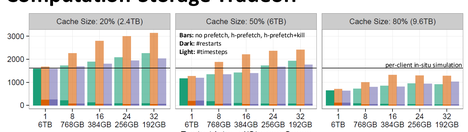
Simulation:
COSMO 2Km resolution
- Output Step: 6GB every 5min (sim. time)
- Restart Step size: 36GB
- Simulated time: 1 week (tot. size: 12TB)

Analysis tools:
Forward: *output steps are accessed in the same time direction as they are produced*
Backward: *inverse direction*
Event tracking: *the analysis tool can change direction according to its analysis*
Random: *fully random access*
ECMWF: *access traces from the European Centre for Medium-Range Weather Forecasts*

Performance Model:



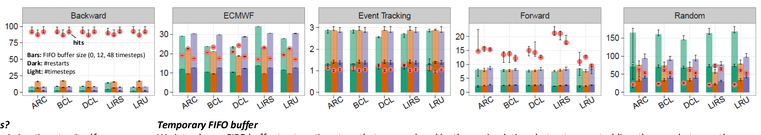
Computation-Storage Tradeoff



Adversary case: 30 analysis tools accessing output steps sequences of random length and starting at random offsets. The output steps are accessed in a forward-in-time trajectory.

Cache Strategy Evaluation:

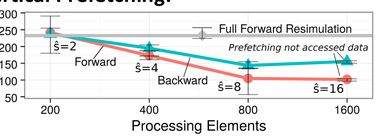
Different cache strategy are analyzed:
ARC: Adaptive Replacement Cache
BCL: Basic Cost-sensitive LRU
DCL: Dynamic Cost-sensitive LRU
LRU: Least Recently Used
LIRS: Low Inter-reference Recency Set



Why is this different from CPU caches?
- The cost of a miss depends on the missing timestep itself.
- A missing timestep *x* leads to the re-simulation of *x mod R* timesteps (i.e., distance from the restart step), that are put in cache.

Temporary FIFO buffer
We introduce a FIFO buffer to store timesteps that are produced by the re-simulations but not requested (i.e., the ones between the restart step and a missing output step). If a subsequent request hits in the FIFO buffer, then the entry is moved into the cache.
Insight: effective for most of the analyzed access patterns. The buffer size needs to be at least one restart interval for backward.

Vertical Prefetching:



Single analysis tool accessing output steps in a forward or backward direction.
SDaVI increases the aggregate simulation bandwidth by launching more simulation in parallel to serve the same analysis tool.

More:

https://digirols.inf.ethz.ch/sdavi

ETH zürich

SC17 Denver, CO hpc connects.