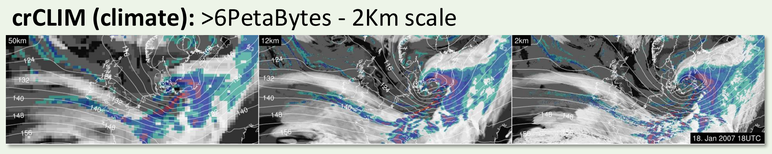
SimFS: A Simulation Data Virtualizing File System Interface

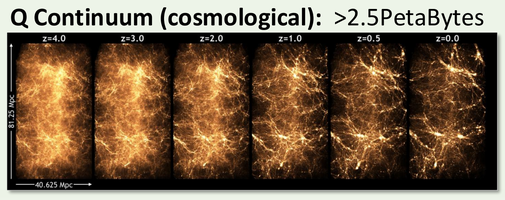
Salvatore Di Girolamo, Torsten Hoefler (advisor)

Big (Simulation) Data Era

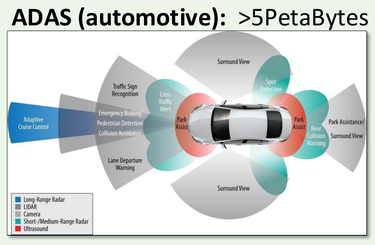
crCLIM (climate): >6PetaBytes - 2Km scale



Q Continuum (cosmological): >2.5PetaBytes



ADAS (automotive): >5PetaBytes



Common Analysis Strategies

On-Disk (offline)

In Situ (online)

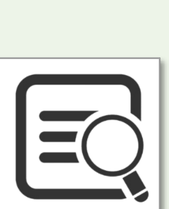
SimFS

SimFS transparently virtualizes the simulation output, relaxing the storage requirements and re-simulating missing data on-demand.

Analysis applications keep working as if all the output data exists (virtualized view).
If the requested data is actually missing, SimFS spawns new simulations to produce it.

SimFS monitors running analyses in order to apply caching and prefetching strategies.

```
SIMFS_Acquire(…, filename, …);
nc_open(filename, …, nid);
// analyze results in filename
nc_close(nid);
SIMFS_Release(…, filename);
```



Simulation

(P)NetCDF, HDF-5, POSIX

SimFS API

DVLib

Control (TCP/IP)

Data Virtualizer

- *Restarts Index*
- *Results Index*
- *Caching Policies*
- *Job Launcher*
- *Analysis Monitoring & Prefetching*

Control (TCP/IP)

(P)NetCDF, HDF-5, POSIX

DVLib

Data Cache

Checkpoints

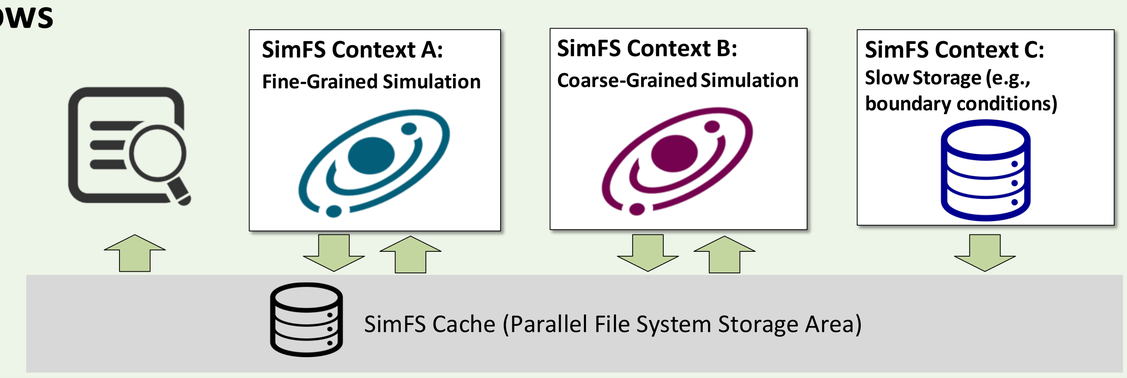
Simulation Contexts

Context A:
Fields: x@2min, y@1h
Restarts: $PROJECT/restarts/2km/
Output: $SCRATCH/data/2km/
Run Script: run_gpu.sh

Context B:
Fields: z@12h
Restarts: $PROJECT/restarts/12km/
Output: $SCRATCH/data/12km/
Run Script: run_cpu.sh

Virtualizing Simulation Workflows

SimFS can be used not only to virtualize single simulations but also to **virtualize entire simulations workflows.**



SimFS Interface

Analysis tools can be interfaced **transparently** or via the **SimFS APIs**

Transparent: Interfacing Legacy Analysis Tools

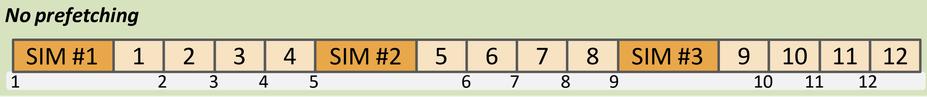
Call	(P)NetCDF	(P)HDF5	ADIOS
open	nc(mpi)_open	H5Fopen	adios_open
read	nc(mpi)_vara_get_type	H5Dread	adios_schedule_read
close	nc(mpi)_close	H5Fclose	adios_close

SimFS APIs: Making Virtualization-Aware Analysis Tools

```
int SIMFS_Acquire(SIMFS_Context context, char * filenames[], int count, SIMFS_Status * status);
int SIMFS_Acquire_nb(SIMFS_Context context, char * filenames[], int count, SIMFS_Status * status, SIMFS_Req * req);
int SIMFS_Release(SIMFS_Context context, char * filename);
int SIMFS_Wait(SIMFS_Req * req, SIMFS_Status * status);
int SIMFS_Test(SIMFS_Req * req, int * flag, SIMFS_Status * status);
```

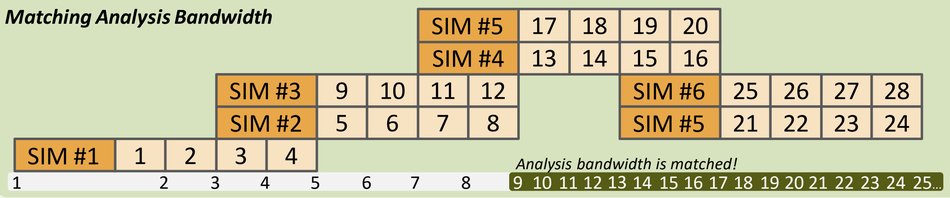
Prefetching Simulation Data

No prefetching



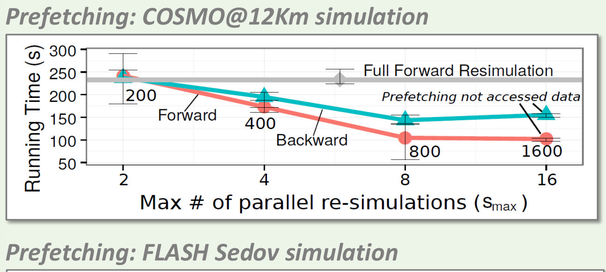
Masking Restart Latency

Matching Analysis Bandwidth

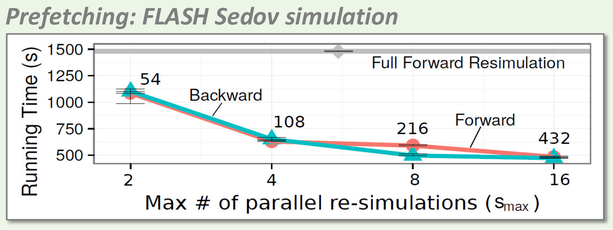


Experiments

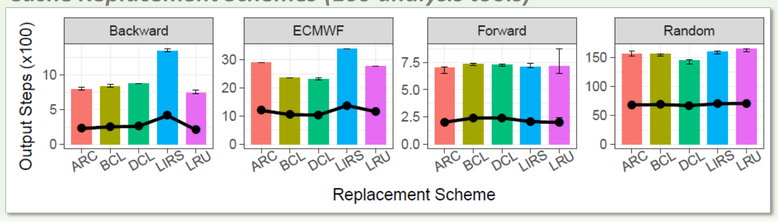
Prefetching: COSMO@12Km simulation



Prefetching: FLASH Sedov simulation



Cache Replacement Schemes (100 analysis tools)

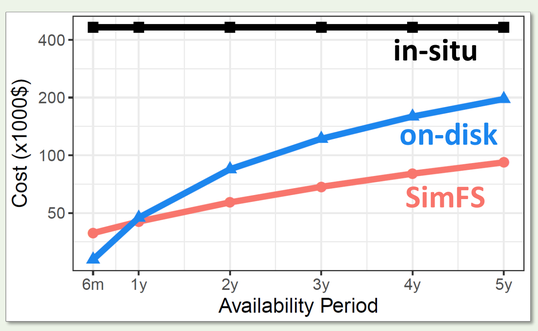


SimFS can improve analysis performance by masking restart overheads and prefetching simulations.

No major differences among the tested caching replacement schemes but cost-aware schemes perform better for real-world scenarios (ECMWF, random)

Cost-Effectiveness

Storing simulation data for long time periods of time is **not cost effective** and, in some cases, **practically impossible**.



Analysis cost over simulation results availablity time

Analysis: 100 analysis tools (forward analysis)
50% execution overlap

Simulation: COSMO (climate simulation)
Time frame: 1 month
Restart files size: 36GiB (total 3TiB)
Data Volume: 6GiB 5min output steps (total 50TiB)

Cost Model

$$C_{\text{on-disk}}(\Delta t) = C_{\text{sim}}(n_o, N) + C_{\text{store}}(n_o, s_o, \Delta t)$$

$$C_{\text{in-situ}}(\Delta t) = \sum_{j}^{m} C_{\text{sim}}(i_j + |\gamma_{\Delta t}(j)|, P)$$

$$C_{\text{SimFS}}(\Delta t) = C_{\text{sim}}(n_o, P) + C_{\text{store}}(n_r, s_r, \Delta t) + C_{\text{store}}(M, s_o, \Delta t) + C_{\text{sim}}(V(\gamma_{\Delta t}), P)$$

Microsoft Azure **Compute:** 2.07$/Node/Hour (NCv2 VM)
Store: 0.06$/GiB/Month (Azure File Share)

ETH zürich

SC18 Dallas, TX | hpc inspires.