Improving the I/O scalability for the next generation of Earth system models: OpenIFS-Cassandra integration as a case study

Authors: Xavier Yepes-Arbós, Pol Santamaria
Co-authors: Kim Serradell, Yolanda Becerra
Contributors: Jesus Labarta, Harald Servat
Introduction

• Earth system models have benefited of the exponential growth of supercomputing power
• This allows to use more complex computational models to find more accurate solutions
• As a consequence, the generated amount of data has grown considerably

Source: Annual Report 2015, ECMWF
Introduction

• However, since the I/O was not significant enough in the past, not much attention was paid to improve it

• Due to this reason, some Earth system models output data using inefficient sequential I/O schemes

• This type of scheme requires a serial process:
  • Gather all data in the master process of the model
  • Then, the master process sequentially writes all data

• This is not scalable for higher grid resolutions, and even less, for future exascale machines

• This is the case of the I/O scheme of the OpenIFS model
OpenIFS model

• OpenIFS is a free and simplified version of the Integrated Forecasting System (IFS), available under a license from the European Centre for Medium-Range Weather Forecasts (ECMWF)

• IFS is a global data assimilation and forecasting system which includes the modelling of the atmospheric composition

• Originally developed for weather forecasting, IFS writes using the GRIB format
GRIB format

• General Regularly-distributed Information in Binary form (GRIB) is a concise data format commonly used for Numerical Weather Prediction (NWP) output

• It is designed to be:
  • Self-describing
  • Compact
  • Portable across computer architectures

• In addition, it offers high performance for I/O operations to meet critical time-to-solution requirements
Post-processing in OpenIFS

• OpenIFS can be also used for climate modelling if coupled into an Earth system model, such as EC-Earth

• However, it is necessary to perform some post-processing, including, but not limited to:
  • Convert GRIB files to netCDF files
  • Regridding
  • Transform spectral fields to grid-point fields
  • Compute diagnostics

• Post-processing turns into an expensive process, which will increase in cost, complexity and size as a result of increasing the resolution
Queries in OpenIFS

- Global-scale simulations over a large period of time produce a huge amount of data
- OpenIFS data is written “as it is” into the storage system
- However, most of the time users are not interested in all data
- When users want to query a subset of the data, they have to post-process (or filter) entire files
- This is a slow and inefficient process difficult to parallelize
Proposed I/O scheme

• As said, OpenIFS has a bottleneck in the sequential I/O scheme which uses a master-workers architecture

• This known issue is being addressed in the ESIWACE2 project

• Our solution is to adapt OpenIFS to perform distributed I/O

• To avoid moving the bottleneck from the model to the storage system, we opted for a distributed masterless database to store and query data: Apache Cassandra

• The main reasons are:
  • Resilience and scalability: to petabytes of data and thousands of nodes
  • Indexation: efficient distributed queries
  • Open-source
Distributed architecture

Traditional architecture

- OpenIFS process 1
- OpenIFS process 2
- OpenIFS process N

  Gather step data

  OpenIFS master process

  Write to grib file

  Centralized storage

Proposed architecture

- OpenIFS process 1
- OpenIFS process 2
- OpenIFS process N

  Cassandra client

  Asynchronous Inserts
Why Cassandra?

Apache Cassandra is a distributed, highly-scalable, and fault-tolerant Key-Value database.

• Pros:
  • Distributed storage, highly scalable
  • Manages coherency, corruption and concurrent access

• Cons: Bad HPC integration
  • IPoverIB
  • Scales up to few cores
  • Client based on thread pool that interferes with simulation
  • Not integrated with HPC queue systems
We tested several data models

1) Represent each grid point as a row to distributed arrays.
   • Pros: Easy to implement
   • Cons: Millions of small insertions → Overhead

\[ \langle \text{lat}_i, \text{lon}_j, \text{llev}_k, \text{ts}_l \rangle, \langle \text{field0}_{ijkl}, \text{field1}_{ijkl}, \ldots \rangle \]

2) Arrays are partitioned and distributed.

\[ \langle \text{ts}_i \rangle, \langle \text{array[field0}_{0..\text{nlat}, 0..\text{nlon}, 0..\text{nlevel}, i, \ldots} \rangle \]

• Our tool Hecuba partitions the arrays
  • Transparent to the client → simplifies its implementation
  • We chose to partition the array into 4K chunks to avoid congesting Cassandra.
OpenIFS configurations evaluated

- OpenIFS 40r1
- Initial conditions: storm Xaver (1 December 2013) → maximum forecast length of 8 days

<table>
<thead>
<tr>
<th></th>
<th>AMIP</th>
<th>HighResMIP</th>
<th>Theoretical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>T255L91</td>
<td>T639L137</td>
<td>T1279L137</td>
</tr>
<tr>
<td># MPI processes</td>
<td>47</td>
<td>141</td>
<td>752</td>
</tr>
<tr>
<td># OpenMP threads</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Time step</td>
<td>2700 seconds</td>
<td>900 seconds</td>
<td>600 seconds</td>
</tr>
</tbody>
</table>
OpenIFS configurations evaluated (2)

<table>
<thead>
<tr>
<th></th>
<th>AMIP</th>
<th>HighResMIP</th>
<th>Theoretical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>T255L91</td>
<td>T639L137</td>
<td>T1279L137</td>
</tr>
<tr>
<td>Forecast length</td>
<td>8 days</td>
<td>8 days</td>
<td>4 days*</td>
</tr>
<tr>
<td>Output frequency</td>
<td>3 hours</td>
<td>3 hours</td>
<td>1 hour</td>
</tr>
<tr>
<td></td>
<td>(4 time steps)</td>
<td>(12 time steps)</td>
<td>(6 time steps)</td>
</tr>
<tr>
<td>Total size GRIB files</td>
<td>9 GB</td>
<td>80 GB</td>
<td>795 GB</td>
</tr>
<tr>
<td>Total size Cassandra</td>
<td>37 GB</td>
<td>355 GB</td>
<td>4224 GB</td>
</tr>
<tr>
<td>(replication factor 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Quota exceeded issue*
HPC platform used

• MareNostrum 4

• Lenovo system composed of:
  • SD530 Compute Racks
  • Intel Omni-Path high performance network interconnect
  • SuSE Linux Enterprise Server

• Compute nodes are equipped with:
  • 2 sockets Intel Xeon Platinum (Skylake) with 24 cores each (48 cores per node)
  • 96 GB of main memory
  • 100 Gbit/s Intel Omni-Path
  • 10 Gbit/s Ethernet

• IBM GPFS file system
OpenIFS performance comparison

![Bar chart comparing execution times for different models and configurations.](chart.png)
Performance Analysis

• Different output size (1 to 4.5 ratio GRIB files vs. Cassandra)
  • Data is stored in Cassandra using double precision but GRIB stores using a scale factor + offset to use variable precision

• Spectral fields transformed to grid-point fields prior to output only with Cassandra

• Cassandra integration with HPC hardware could be improved

• Slightly slower than files but the benefits are on queries:
  • Efficient filtering (post-processing operations) and sampling
  • Distributed computing
  • Concurrent access while data is generated
BSC-Intel collaboration

• Apache Cassandra adapted by Intel to run on the Intel(R) Optane(TM) DC Persistent Memory
  • Will boost the performance of both OpenIFS and post-processing applications

![Benchmarks on Cassandra](image)
Common post-process operations

- **Regridding**
  - Reduced Gaussian to rectangular

- Temporal queries over a period of time: average, maximum, minimum, cumulate, etc

- Spatial queries: regional domain extraction, collapsing in the vertical direction or in the horizontal, etc
Post-process workflow

- Typical approach for implementing queries in the Earth Science domain:
  - Bash or Python script that manipulates the OpenIFS output files. Filters entire files, and each post-process step generates new files.

- Our proposal: parallel post-processing using PyCOMPSs and Hecuba
  - PyCOMPSs offers an automatic parallelization of sequential code
  - Hecuba offers a transparent access to persistent objects in Cassandra
Conclusions

• Room for improvement: Cassandra’s C++ Client makes use of threads, which conflict with MPI

• Potential benefits for the post-processing operations
  • Parallel execution
  • Easy to implement
  • Possibility of using standard libraries to manipulate the data (NumPy)

• Intel’s Cassandra is promising: we expect a performance boost thanks to the Intel(R) Optane(TM) DC Persistent Memory
Future work

• Understand the impact on OpenIFS of the Cassandra client threads

• Complete and evaluate the distributed queries

• Write data in double, single or smaller precision depending on the user needs

• Evaluate OpenIFS taking advantage of Intel’s Cassandra in a distributed environment
Thank you

xavier.yepes@bsc.es
pol.santamaria@bsc.es