Maestro Project Introduction

François Tessier
Swiss National Supercomputing Centre, ETH Zurich, Lugano, Switzerland

PADAL Workshop 2019
Bordeaux, France
Context

Complex workflows or frameworks in various scientific domains have increasing I/O needs

<table>
<thead>
<tr>
<th>Institution</th>
<th>Scientific domain</th>
<th>Workflows</th>
<th>Data size (real &amp; projection)</th>
</tr>
</thead>
<tbody>
<tr>
<td>European Centre for Medium-Range Weather Forecasts (ECMWF)</td>
<td>Weather Forecast</td>
<td>Ensemble forecasts, data assimilation,…</td>
<td>25PB/year (2025: 350PB/year)</td>
</tr>
<tr>
<td>Paul Scherrer Institute (PSI)</td>
<td>Synchrotron imaging</td>
<td>X-ray spectroscopy, high resolution microscopy,…</td>
<td>10-20PB/year</td>
</tr>
<tr>
<td>Cherenkov Telescope Array (CTA)</td>
<td>Astrophysics</td>
<td>Gamma Rays &amp; Cosmic Sources,…</td>
<td>25PB/year</td>
</tr>
</tbody>
</table>

● Workloads with specific needs of data movement
  ○ Big data analysis, machine learning, checkpointing, in-situ, co-located processes, …
  ○ Multiple data access patterns (model, layout, data size, frequency)
Context

- But the ratio “I/O performance” / “computing power” is decreasing!

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name, Location</td>
<td>BlueGene/L, USA</td>
<td>Sunway TaihuLight, China</td>
<td>N/A</td>
</tr>
<tr>
<td>Theoretical perf.</td>
<td>596 TFlops</td>
<td>125,436 TFlops</td>
<td>× 210</td>
</tr>
<tr>
<td>#Cores</td>
<td>212,992</td>
<td>10,649,600</td>
<td>× 50</td>
</tr>
<tr>
<td>Total Memory</td>
<td>73,728 GB</td>
<td>1,310,720 GB</td>
<td>× 17.7</td>
</tr>
<tr>
<td>Memory/core</td>
<td>346 MB</td>
<td>123 MB</td>
<td>÷ 2.8</td>
</tr>
<tr>
<td>Memory/TFlop</td>
<td>124 MB</td>
<td>10 MB</td>
<td>÷ 12.4</td>
</tr>
<tr>
<td>I/O bw</td>
<td>128 GBps</td>
<td>288 GBps</td>
<td>× 2.25</td>
</tr>
<tr>
<td>I/O bw/core</td>
<td>600 kbps</td>
<td>27 kbps</td>
<td>÷ 22.2</td>
</tr>
<tr>
<td>I/O bw/TFlop</td>
<td>214 MBps</td>
<td>2.30 MBps</td>
<td>÷ 93.0</td>
</tr>
</tbody>
</table>

- Mitigating the I/O bottleneck from an hardware perspective leads to an increasing complexity and a diversity of the multiple tiers
  - Node-local storage (PCIe, SATA)
  - Burst buffers like Cray DataWarp, DDN Infinite Memory Engine
But the ratio “I/O performance” / “computing power” is decreasing!

Mitigating the I/O bottleneck from an hardware perspective leads to an increasing complexity and a diversity of the multiple tiers
- Node-local storage (PCIe, SATA)
- Burst buffers like Cray DataWarp, DDN Infinite Memory Engine

<table>
<thead>
<tr>
<th>System Specs</th>
<th>TITAN</th>
<th>SUMMIT</th>
<th>FRONTIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak Performance</td>
<td>27 PF</td>
<td>200 PF</td>
<td>&gt;1.5 EF (× 7.5)</td>
</tr>
<tr>
<td>Storage</td>
<td>32 PB, 1 TB/s Lustre file-system</td>
<td>250 PB, 2.5 TB/s GPFS</td>
<td>2-4x performance and capacity of Summit’s I/O subsystem. Frontier will have near node storage like Summit.</td>
</tr>
</tbody>
</table>

Source: https://www.olcf.ornl.gov/frontier/
Hardware Architecture Examples: Summit

Source: https://www.olcf.ornl.gov

Source: https://fuse.wikichip.org
Today’s Shortcomings

**Data Awareness**

- HPC Software stack focusing on data processing
  - Optimised for filling the processing pipelines
  - Provide means for leveraging parallelism

- Lacking basic data handling at various levels of the stack
  - Lacking functionality for controlling data handling
  - Lacking (unified) semantics for guiding data transport

**Memory Awareness**

- Missing information about available memory/storage hardware and its characteristics
  - Lacking ability for making data transport decisions
  - Missing information leads to hardware-neutral decisions

- Challenging variety of data access methods
  - Example storage class memory: Block store, file system, object storage

- This becomes more critical with deeper memory and storage hierarchies
Maestro

- Maestro will build a **data and memory-aware middleware framework** that addresses the ubiquitous problems of data movement in complex memory hierarchies that exist at multiple levels of the HPC software stack.

- 3-year European project, started in September 2018, involving partners from academia and industry
Design of the Maestro middleware

- Workflow description
- Architecture
- Workflow Translator
- Workflow manager
- Workload manager
- Dynamic provisioning
- Maestro-enabled tasks
  - Data Manager Tasks
    - Task 1
    - Task 2
    - Task 3

Existing component
MAESTRO component
Task (job)
On-demand data manager
Design of the Maestro middleware

Architecture
- Lustre parallel file-system
- Near-compute SSD nodes (DataWarp)
- Memory hierarchy

Maestro attributes for Task 1
- `maestro.workload.requirements.persistency: workflow_lifetime`
- `maestro.workload.characteristic.metadata_intensive`
- `maestro.workload.data_management: 'file'`

Translator
- Task 1 close to intermediate storage nodes
- Dynamically provisioned BeeGFS (extra task)
- New graph of dependencies
Design of the Maestro middleware

Workflow description

Architecture

MAESTRO component

Existing component

Workflow description extension

Workflow Translator

Dynamic provisioning

Workflow manager

Workload manager

Maestro-enabled tasks

On-demand data manager

Data Manager Tasks

Task 1

Task 2

Task 3

Workload manager

Resource allocation requests

Core Data Object API: declare, give, take, dispose

Unified Memory-storage API: alloc, free, store, load, flush

Maestro Data Management

CDO POOL

Scope Object

Maestro Data Transformation

Mem

DRAM

HBM

NVRAM

SSD

PFS

00101101010110

dole, give, take, dispose

Maestro System Model

CDO

Sys

Cost

10
Design of the Maestro middleware

- **Workflow description**
- **Workflow Translator**
- **Workflow manager**
- **Workload manager**
- **Dynamic provisioning**
- **Maestro-enabled tasks**
  - Data Manager Tasks
  - Task 1
  - Task 2
  - Task 3

**Existing component**
- MAESTRO component
- Task (job)
- On-demand data manager

**Workflow description extension**

**Architecture**

**Resource allocation requests**

**Core Data Object API:**
- declare, give, take, dispose

**Task 1**
- CDO give

**Task 2**
- CDO take

**Task 3**
- CDO give

**Maestro Data Management**
- CDO POOL
- CDO

**Maestro Data Transformation**
- Scope Object
- CDO

**Maestro System Model**
- CDO → Sys → Cost

**Unified Memory-storage API:**
- alloc, free, store, load, flush

**Resources**
- DRAM
- HBM
- NVRAM
- SSD
- PFS
Design of the Maestro middleware

**CDO (Core Data Object)**
It is at the heart of Maestro’s design and is used to encapsulate data and metadata. Supports dependencies.

**GIVE**
Applications give CDOs to the management pool. Maestro manages the data.

**TAKE**
When an application takes a CDO, Maestro relinquishes all control of the data.

**SCOPE OBJECT**
Captures information about scope, size, access relations and schedules of the data to enable efficient movement and/or transformation.

**MAESTRO SYSTEM MODEL**
Computes the cost of moving, transforming or copying data a CDO.

**SYS**
Interface to every memory level, enabling core functionality of that memory.
Design of the Maestro middleware

**Hardware**
- Memory, storage

**Software**
- Semantics
  - Graph of Mem objects
  - Memory/storage tier
  - Access interface

**CDO**
- Unit of data in Maestro
- Given and taken to/from Maestro
- Relations, dependencies

**Task 1**
- CDO give

**Task 2**
- CDO take

**Task 3**
- CDO give

Core Data Object API: declare, give, take, dispose

Maestro Data Management
- CDO POOL
- Scope Object
- Maestro Data Transformation

Maestro System Model
- CDO → Sys → CDO

Unified Memory-storage API: alloc, free, store, load, flush

- DRAM
- HBM
- NVRAM
- SSD
- PFS

Resources

Semantics
- Graph of Mem objects
- Memory/storage tier
- Access interface

**Mem**
- Extent (blob length, idx space)
- Metadata

**Scope Object**
- Unit of data in Maestro
- Given and taken to/from Maestro
- Relations, dependencies

**Sys**
Co-Design Applications

- IFS numerical weather prediction system
  - Complex data processing and simulation system with multiple data producers and consumers

- Computational Fluid Dynamics plus in-situ analysis
  - Pipeline coupling multiple simulations plus data post-processing

- Electronic structure calculation library SIRIUS
  - Simulations involving GPU acceleration

- Global Earth Modelling system TerrSysMP
  - Coupled simulations
Today’s bottlenecks

- Data movement between forecast stages and product generation
- Irregular archiving of output from research workflows
Example: Weather Prediction Workflow

Today’s bottlenecks

- Data movement between forecast stages and product generation
- Irregular archiving of output from research workflows
Summary and Outlook

- Today’s HPC (and HPDA) solutions lack data and memory awareness
- Maestro will develop a data and memory aware middleware
  - Abstractions based on data objects
  - Memory-aware data transport and placement in middleware
- Tag tasks with data-related information, tag data with metadata (ownership, location, size, and so on)
- Open for providing early access to technology

Project Schedule

- Requirements definition completed in August 2019
- Core design fully specified by April 2020
- Start application demonstration this autumn
- Project completion in August 2021
Conclusion

Thank you for your attention!

francois.tessier@cscs.ch

Acknowledgment

- This work is part of the MAESTRO EU Project
- 3-year European project, started in September 2018
- Middleware library that automates data movement across diverse memory systems
- https://www.maestro-data.eu/