MANAGING COMPOSABLE DISAGGREGATED INFRASTRUCTURE WITH OFA SUNFISH

Christian Pinto

Staff Research Scientist, IBM Research Europe
Co-chair, OpenFabrics Alliance Management Framework Workgroup
CONTRIBUTORS

- Michele Gazzetti (IBM Research Europe)
- Phil Cayton (Intel)
- Russ Herrell (Hewlett Packard Enterprise)
- Michael Aguilar (Sandia National Labs)
- Brian Pan (H3 Platform)
- Ziyan Zhuang (H3 Platform)
- Jin Hase (Fsas Technologies Inc.)
- Naoki Oguchi (Fsas Technologies Inc.)
- Network(fabric)-disaggregated infrastructure becoming the state-of-the-art
- No common fabric manager interface or fabric model available to link applications with remote resources
- Administrators asked to manage an increasing heterogenous fabrics infrastructure
- Difficult to automate because different fabrics require different optimizations
OUR FOCUS SO FAR

Clients

Composability and Resources Managers

Sunfish Core Services

Sunfish Agents

Hardware Managers

Application Domain

App driven system reconfiguration

System composition, Systems update

Infrastructure management

Composability Layer

Data Store

Fabric Resources Monitoring

Composition Policies

Resource Managers (e.g., Compute, FAM, Storage, Fabric)

Events & Logs

Sunfish Services

- Resource Inventory
- RF tree management
- Resource Configuration
- Fabric Configuration
- Authentication
- Access Control
- Events & Logs

Sunfish Framework

Redfish / Native API Translation
RedFish API
Vendor Native API

© OpenFabrics Alliance
OUR FOCUS SO FAR

**Clients**

**Composability and Resources Managers**
- Fabric Resources Monitoring
- Composition Policies
- Resource Managers (e.g., Compute, FAM, Storage, Fabric)
- Events & Logs

**Sunfish Core Services**
- Resource Inventory
- RF tree management
- Resource Configuration
- Fabric Configuration
- Authentication
- Access Control

**Sunfish Services**
- Events & Logs

**Sunfish Agents**
- CXL Agent
- NVMeoF Agent
- InfiniBand Agent
- Additional Agents

**Hardware Managers**
- CXL Manager(s)
- NVMe Manager
- Infiniband Manager
- Additional Manager

**Additional Managers**
- CXL HW
- NVMe HW
- Infiniband HW
- ??? HW

**Redfish / Native API Translation**

**Redfish API**

**Vendor Native API**

**Sunfish Framework**

© OpenFabrics Alliance
OUR FOCUS SO FAR

Clients

Composability and Resources Managers
- Fabric Resources Monitoring
- Composition Policies
- Resource Managers (e.g., Compute, FAM, Storage, Fabric)
- Events & Logs

Sunfish Core Services
- Resource Inventory
- RF tree management
- Resource Configuration
- Fabric Configuration
- Authentication
- Access Control
- Events & Logs

Sunfish Agents
- CXL Agent
- NVMeoF Agent
- InfiniBand Agent
- Additional Agents

Hardware Managers
- CXL HW
- NVMe HW
- InfiniBand HW
- Additional Manager

Application Domain
- App driven system reconfiguration

Administration Domain
- Systems composition, Systems update
- Infrastructure management

Sunfish Framework

Composability Layer

API

Redfish / Native API Translation

Vendor Native API

© OpenFabrics Alliance
WHY SUNFISH?

Compute Nodes
- Sys1
- Sys2
- Sys3
- Sys4
- Sys5

Root

Systems
- Sys1
- Sys2
- Sys3
- Sys4
- Sys5

Fabrics
- CXL
- NVMeoF
- PCIe

Interconnect Fabrics
- Mem
- Acc
- NVMe
- GPU

Resource Pools
Only composed systems are of interest to the client. Sunfish abstracts the fabric and connection details out, providing the logical view of a "classic" computer system.
User requests pod using GPU to K8s
If there is no adequate GPU, Kubernetes requests resource to CDI to attach GPU to a worker node
Kubernetes maps a scheduled POD to the attached GPU
CDI follows Sunfish architecture
**Conventional operation**
The maximum number of GPUs are always running assuming maximum load. In the example below, 10 GPUs are always powered on.

<table>
<thead>
<tr>
<th>Load (Data Traffic)</th>
<th>Time</th>
<th>Number of GPUs powered on</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Problem**
Power consumption is constantly high (e.g. In case of H100, 49W at idle per GPU)

**Solution by CDI**
CDI automatically increases/decreases the number of GPUs based on load. Unused GPUs are returned to the resource pool to power down and save power.

<table>
<thead>
<tr>
<th>Load (Data Traffic)</th>
<th>Time</th>
<th>Number of GPUs running</th>
<th>Number of GPUs not running (power off)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Improvement**
- 6% reduction in power consumption by powering down idle GPUs
- ~25% of total GPU cycles are then available for other workloads
SUNFISH IS FINALLY OUT!

The OFMF Workgroup is happy to announce the first official release of the Sunfish Framework

- Official documentation
- Reference software implementation
Requirements and normative references for implementing a fully compliant Sunfish Framework, Hardware Agent and Client

- Sunfish framework components design and interactions description
- Interactions between Sunfish and Hardware Agents
- Hardware Agents lifecycle management (registration, failover, etc.)
- Redfish/Swordfish schema objects adopted
- Additions to Redfish schema
- Sunfish specific Redfish modeling requirements (e.g., CXL Fabric Attached Memory)
- **Reference Sunfish Core Library**
  - Code: [https://github.com/OpenFabrics/sunfish_library_reference](https://github.com/OpenFabrics/sunfish_library_reference)
  - Implements the Sunfish core services as a python library
    - RedFish tree management
    - Interactions with Hardware Agents
    - Events brokerage

- **Reference Sunfish Server**
  - Code: [https://github.com/OpenFabrics/sunfish_server_reference](https://github.com/OpenFabrics/sunfish_server_reference)
  - RESTful API for the Sunfish core library

- **Reference Sunfish Hardware Agent**
  - Work on CXL Hardware Agent in progress
  - Agent API to Sunfish Server being developed for CXL FAM
  - Agent backend being developed for CXL fabric mock-ups
FIRST HARDWARE AGENT FOR CXL MEMORY

H3 Fabric Manager
+ Xconn CXL Switch
+ E3.S CXL Memory Module
FIRST HARDWARE AGENT FOR CXL MEMORY

Redfish Agent Logical View
FIRST HARDWARE AGENT FOR CXL MEMORY

CXL Memory Solution

- 24 E3.S 2T Memory
- Expandable CXL ports
- Management port of mCPU for Redfish agent
- CXL host ports
DEMONSTRATION

Systems
- Sys1
- Sys2
- Sys3

Fabrics
- CXL (H3 Platform)
- Endpoints
- Mem
- S1T11
- S1I3

Root

PCle (Liqid)
- PCIe
- Endpoints
- pcpu1
- gpu1

Connection to be established
1. Attach GPU or CXL memory to hosts
The Sunfish community is rapidly growing, and we are targeting further hardware vendors for creating an ecosystem of agents.

Focus on integrating with clients (e.g., Kubernetes, Flux, etc.) to demonstrate the value of a single API approach.

Sunfish will be at SC’24 in Atlanta, GA

Join the community:
- Contributions welcome:
  - Workload managers integration
  - Parallel computing libraries integration
  - More agents for real disaggregated hardware products
- How to join
  - Meeting weekly on Fridays @7am Pacific Time
    - [https://www.openfabrics.org/my-calendar/#mc_calendar_05_2802-calendar-details-my-calendar](https://www.openfabrics.org/my-calendar/#mc_calendar_05_2802-calendar-details-my-calendar)
  - Join the Mailing list:
    - [https://lists.openfabrics.org/mailman/listinfo/ofmfwg](https://lists.openfabrics.org/mailman/listinfo/ofmfwg)
- Reach out for information
  - Christian Pinto: christian.pinto@ibm.com
  - Michael Aguilar: mjaguil@sandia.gov
THANK YOU

Christian Pinto

IBM Research Europe