|  |  |
| --- | --- |
| Use-Case Description  | Composability Manager--Assemble System |
| Actors  | Composability Manager, Sunfish, Free Pool, Active Pool, Janusgraph database, Resource Control Operations, Authorization Block, Composition Policies |
| Description  | Assemble a composed HPC node using resources out of the Free Pool |
| Input Data | Client requirements: CPU architecture, Memory Device Type, Storage Capacity, Storage Type, Accelerator Type, Network Interface Types, best choice for resource selection from Composition Decisions and Policies, available resources from Janusgraph |
| Pre Conditions | OFMF contains a Free pool of Resources, network Agents active |
| Post Conditions | Composed Turing Compatible System from Free Pool, Active Pool incremented by Composed Resources |
| Trigger | Client request for fully Composed Resources |
| Normal Flow | * Is this request for a dynamic expansion to a running allocated server?
* Is this request for dynamic expansion to an unallocated server for batch job allocation?
* Receive Client Requirements
* g.V().has (‘<property>’ to provide appropriate resources from Janusgraph
* GET current Free Pool resources from the Janusgraph database
* Get best choice for resource selection from Decisions and Policies Block
* Create a framework package of allocation requirements, using the Client Requirements, the available Free Pool Resources, and input from the Decisions and Policies Block
	+ Create JSON
		- CPU
			* Type of CPU(s)
			* Quantity of CPU(s)
		- Memory
			* Type of Memory
			* Amount of Memory
		- Storage
			* Type of Storage
		- Resource endpoints
			* What network links are available?
			* What networks?
			* Aggregated?
* Associate Components with links into the Sunfish-Redfish/Swordfish Tree
* POST Constrained Composable JSON to Sunfish
	+ Post of Free Resources to /redfish/v1/CompositionService/ResourceZones and /redfish/v1/CompositionService/ResourceBlocks
* POST used resources to Sunfish-Active Resources
* g,addE(‘property’).from vertex to vertex path
* Update vertex and edge information in the Decisions and Policies Block

Return success |
| Alternate Flow 1 | * Receive Client Requirements
* .V().has (‘<property>’ to provide appropriate resources from Janusgraph
* Get current Free Pool resources from the Janusgraph database
* Resources don’t exist to fulfill requirements

Return failure |
| Alternatie Flow 2 | * Receive Client Requirements
* .V().has (‘<property>’ to provide appropriate resources from Janusgraph
* Get current Free Pool resources from the Janusgraph database
* Resources don’t exist to fulfill requirements, locally
* Get best choice for resource selection from Decisions and Policies Block
* Create a framework package of JSON Constrained requirements, using the Client Requirements and the available Free Pool Resources
* Warning to the client that the requested resources are remote
	+ Create JSON
		- CPU
			* Type of CPU(s)
			* Quantity of CPU(s)
		- Memory
			* Type of Memory
			* Amount of Memory
		- Storage
			* Type of Storage
		- Resource endpoints
			* What network links are available?
			* What networks?
			* Aggregated?
* Associate Components with links into the Redfish Tree
* POST Constrained Composable JSON to Sunfish
	+ Post of Free Resources to /redfish/v1/CompositionService/ResourceZones and /redfish/v1/CompositionService/ResourceBlocks
* POST used resources to Active Resources
* Return success
 |