Futuristic Computational Model in the Era of NVRAM

...

Arpit Gupta
Advisor: Debadatta Mishra

Introduction

- Data intensive applications demand memory capacity
- DRAM can no longer provide the capacity needed

Introduction

- Data intensive applications demand memory capacity

- DRAM can no longer provide the capacity needed

- Non Volatile Memory (NVM) technology
  - higher density

Introduction

- Data intensive applications demand memory capacity
- DRAM can no longer provide the capacity needed
- Non Volatile Memory (NVM) technology
  - higher density
  - slower

Introduction

- Data intensive applications demand memory capacity
- DRAM can no longer provide the capacity needed
- Non Volatile Memory (NVM) technology
  - higher density
  - slower

Cannot Replace DRAM entirely

Introduction

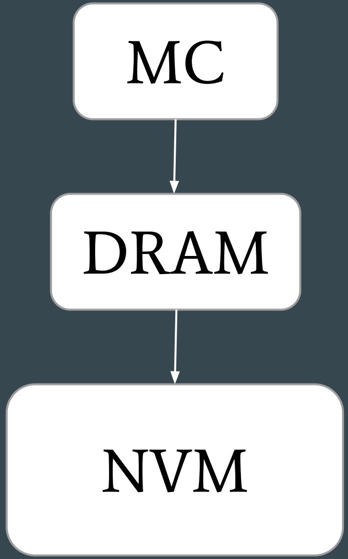
- Data intensive applications demand memory capacity
- DRAM can no longer provide the capacity needed
- Non Volatile Memory (NVM) technology
  - higher density
  - slower

Cannot Replace DRAM entirely

Solution: **Hybrid Memory Systems**

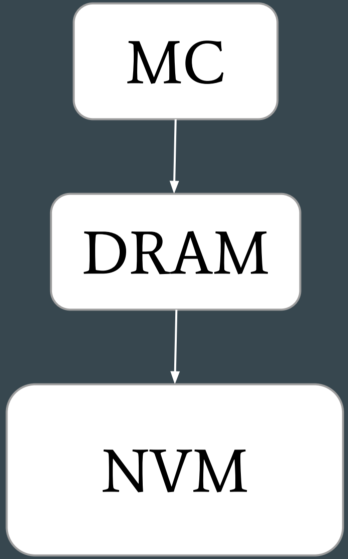
Hybrid Memory Systems (HMS)

- DRAM as a cache for the NVM
    - Capacity Loss



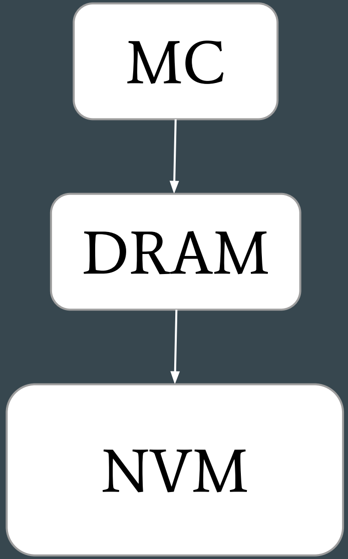
Hybrid Memory Systems (HMS)

- DRAM as a cache for the NVM
    - Capacity Loss
    - Bandwidth Loss



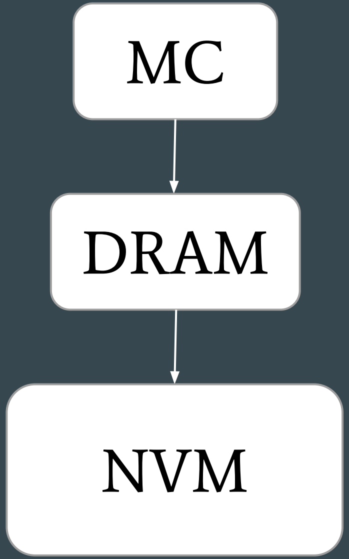
Hybrid Memory Systems (HMS)

- DRAM as a cache for the NVM
  - Capacity Loss
  - Bandwidth Loss
  - DRAM need subtle changes



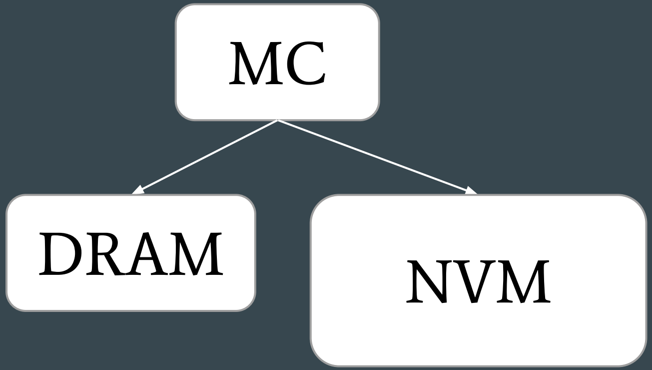
Hybrid Memory Systems (HMS)

- DRAM as a cache for the NVM
  - Capacity Loss
  - Bandwidth Loss
  - DRAM need subtle changes
  - No OS changes required



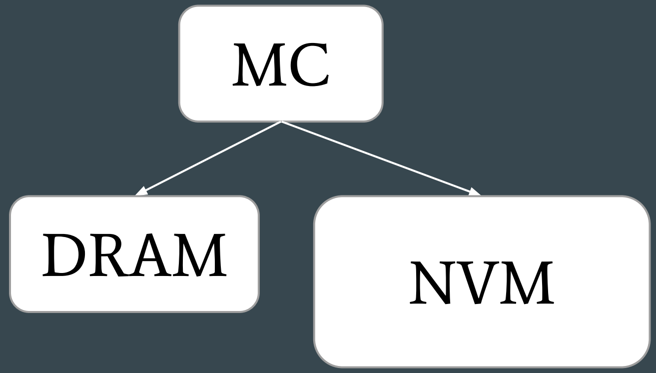
Hybrid Memory Systems (HMS)

- DRAM and NVM share a flat address space
    - More Capacity



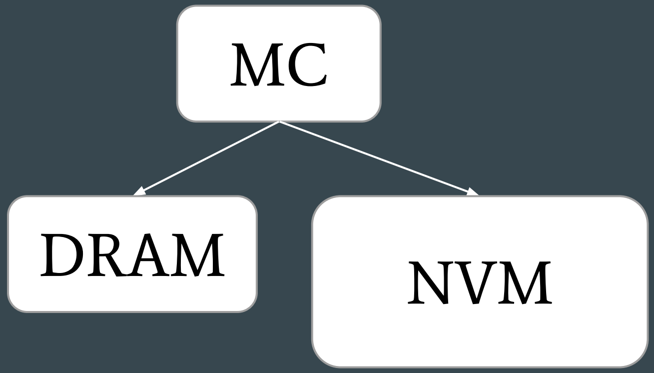
Hybrid Memory Systems (HMS)

- DRAM and NVM share a flat address space
  - More Capacity
  - More Bandwidth



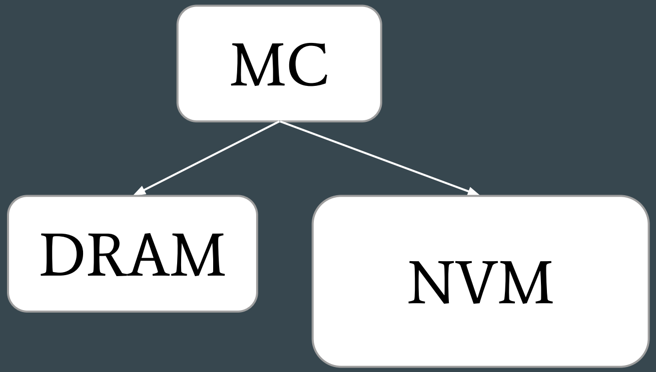
Hybrid Memory Systems (HMS)

- DRAM and NVM share a flat address space
  - More Capacity
  - More Bandwidth
  - No DRAM changes



Hybrid Memory Systems (HMS)

- DRAM and NVM share a flat address space
  - More Capacity
  - More Bandwidth
  - No DRAM changes
  - **OS support is helpful**



Existing Work

Surveys show that hardware and software issues needs to be tackled for successful integration of NVMs

Existing Work

Surveys show that hardware and software issues needs to be tackled for successful integration of NVMs

- Emulation of NVMs

Existing Work

Surveys show that hardware and software issues needs to be tackled for successful integration of NVMs

- Emulation of NVMs
- Programming Interface to NVM

Existing Work

Surveys show that hardware and software issues needs to be tackled for successful integration of NVMs

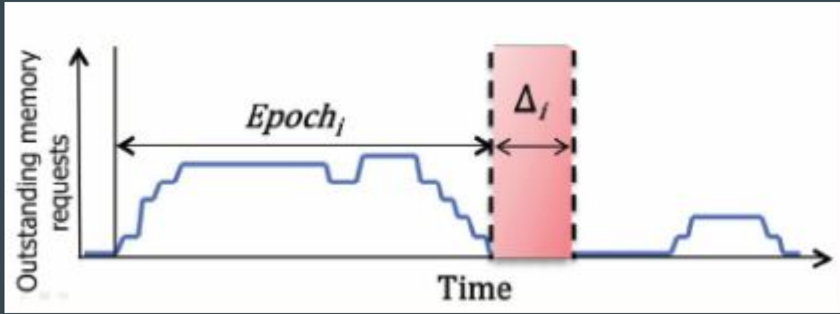
- Emulation of NVMs
- Programming Interface to NVM
- Efficient File System

Existing Work

Surveys show that hardware and software issues needs to be tackled for successful integration of NVMs

- Emulation of NVMs
- Programming Interface to NVM
- Efficient File System
- Operating Systems

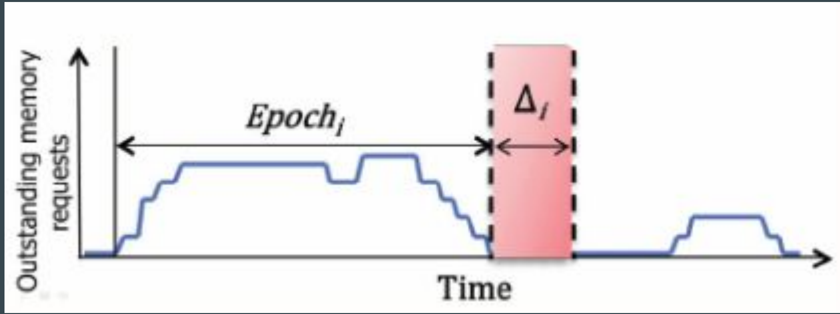
Emulation: Quartz



- Latency Model
  - Dynamically injects software created delays

$$\Delta_i = M_i.(NVM_{lat} - DRAM_{lat})$$

Emulation: Quartz



- Latency Model
  - Dynamically injects software created delays

$$\Delta_i = M_i.(NVM_{lat} - DRAM_{lat})$$

- Bandwidth Model
  - DRAM thermal control feature

Programming Model: Mnemosyne

- Programming interface to Non Volatile Memory

Programming Model: Mnemosyne

- Programming interface to Non Volatile Memory
- Provides consistency in case of power failure

Programming Model: Mnemosyne

- Programming interface to Non Volatile Memory
- Provides consistency in case of power failure
- Allows dynamic memory allocation to NVM
    - Ensures consistency

Programming Model: Mnemosyne

- Programming interface to Non Volatile Memory
- Provides consistency in case of power failure
- Allows dynamic memory allocation to NVM
    - Ensures consistency
    - Resistant to memory leaks

Operating Systems: Twizzler

- Persistent Object Access

Operating Systems: Twizzler

- Persistent Object Access
- Security
    - Objects can be composed => pose a threat (isolation)

Operating Systems: Twizzler

- Persistent Object Access
- Security
  - Objects can be composed => pose a threat (isolation)
- Persistent Kernel Space

Operating Systems: Twizzler

- Persistent Object Access
- Security
  - Objects can be composed => pose a threat (isolation)
- Persistent Kernel Space
- **Lack of Implementation Details and Incomplete**

Object Model

- Abstract binary files to object

Object Model

- Abstract binary files to object
  - Has its own code and data

Object Model

- Abstract binary files to object
  - Has its own code and data
- Provide processes, handles to these objects

Object Model

- Abstract binary files to object
  - Has its own code and data
- Provide processes, handles to these objects
- Process can call function from these objects
  - Carry out computation

Object Model

- Abstract binary files to object
  - Has its own code and data
- Provide processes, handles to these objects
- Process can call function from these objects
  - Carry out computation
- Multiple Processes can attach to same object

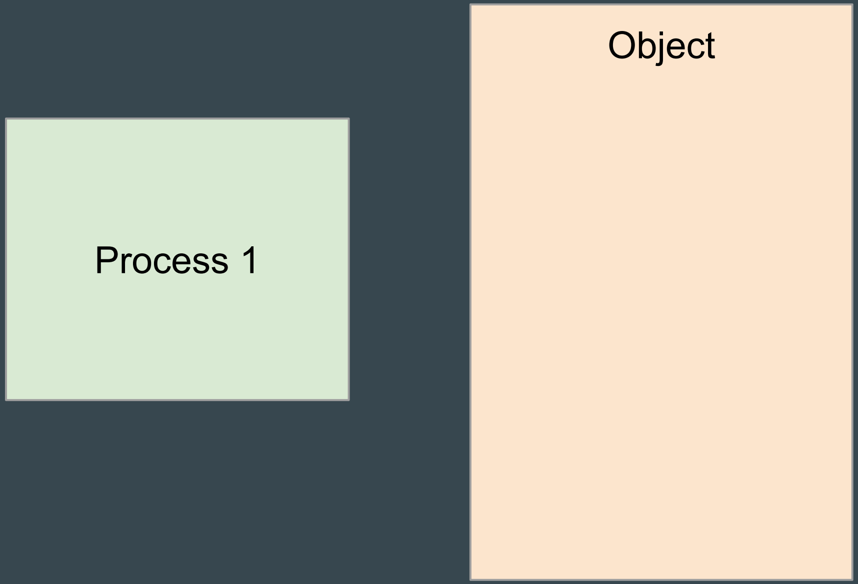
Object Model

- Abstract binary files to object
  - Has its own code and data
- Provide processes, handles to these objects
- Process can call function from these objects
  - Carry out computation
- Multiple Processes can attach to same object
  - Pipeline Processing

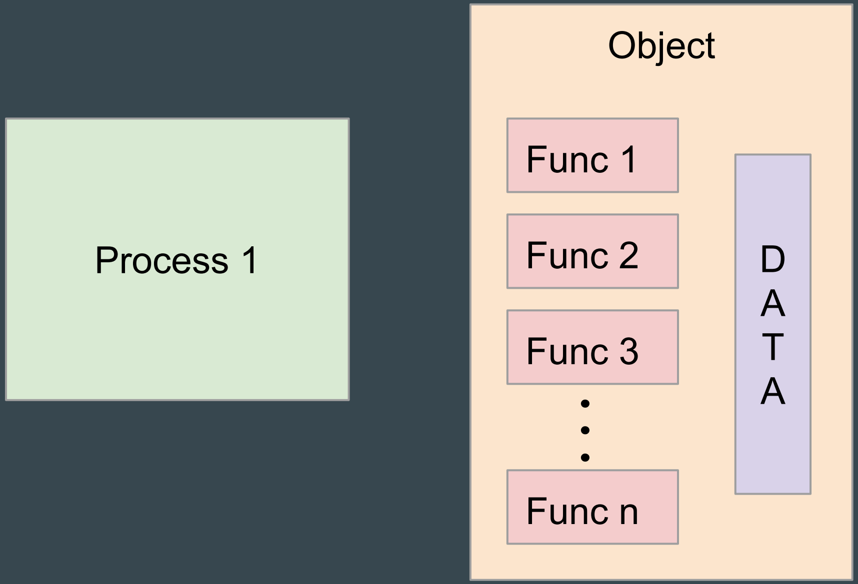
High Level View

Process 1

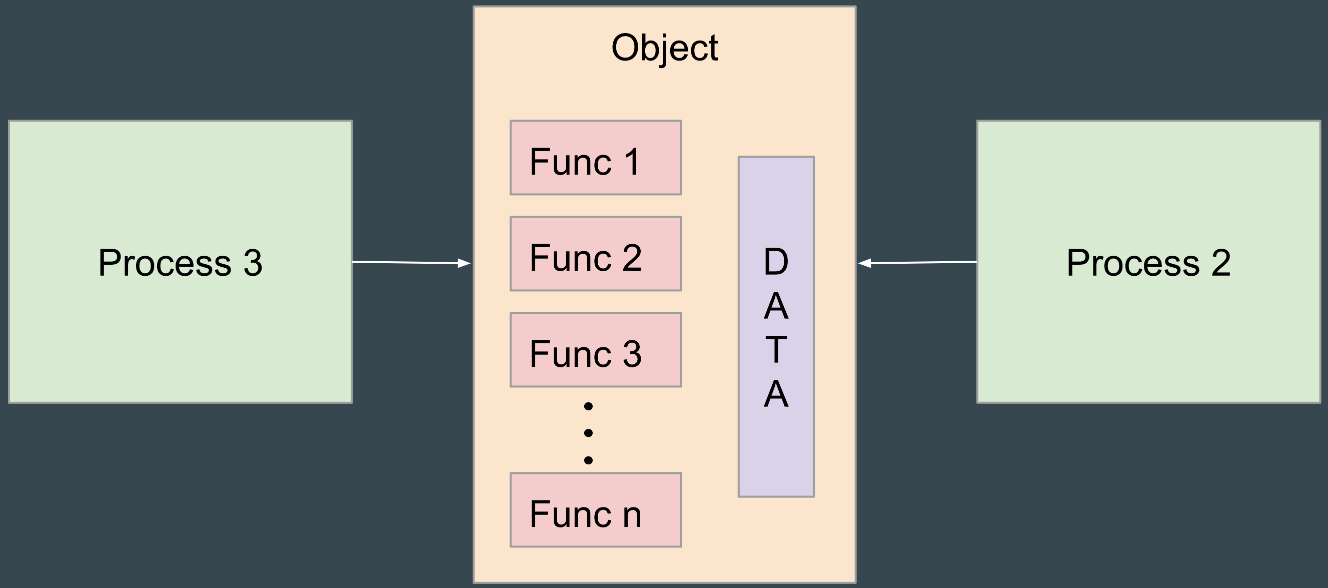
High Level View



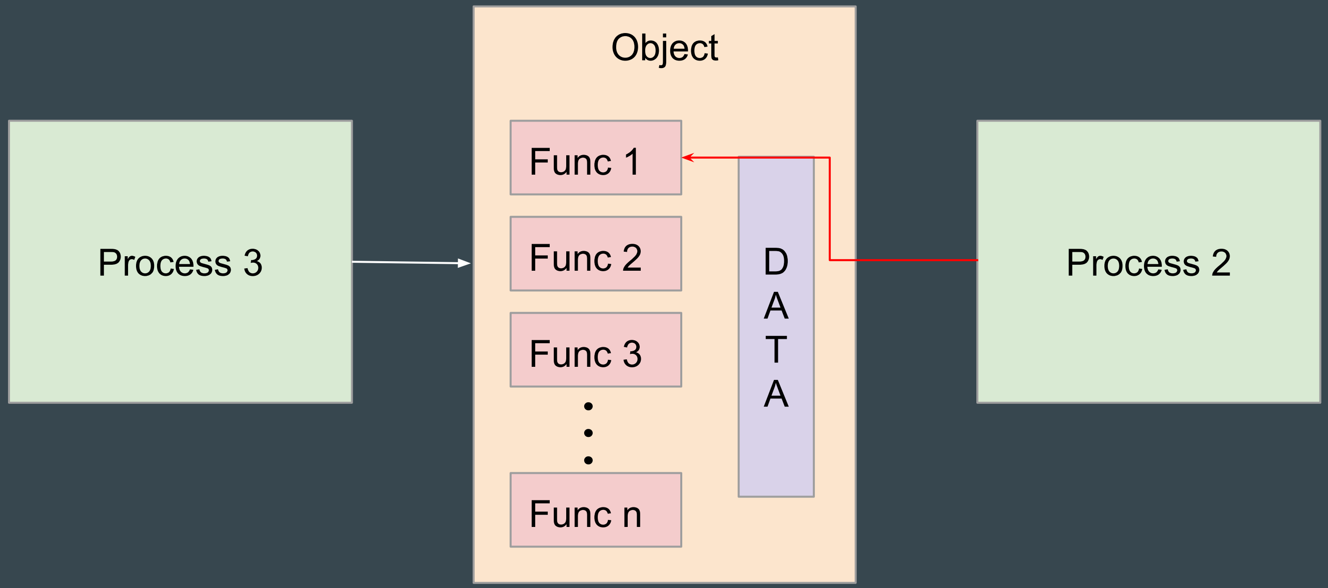
High Level View



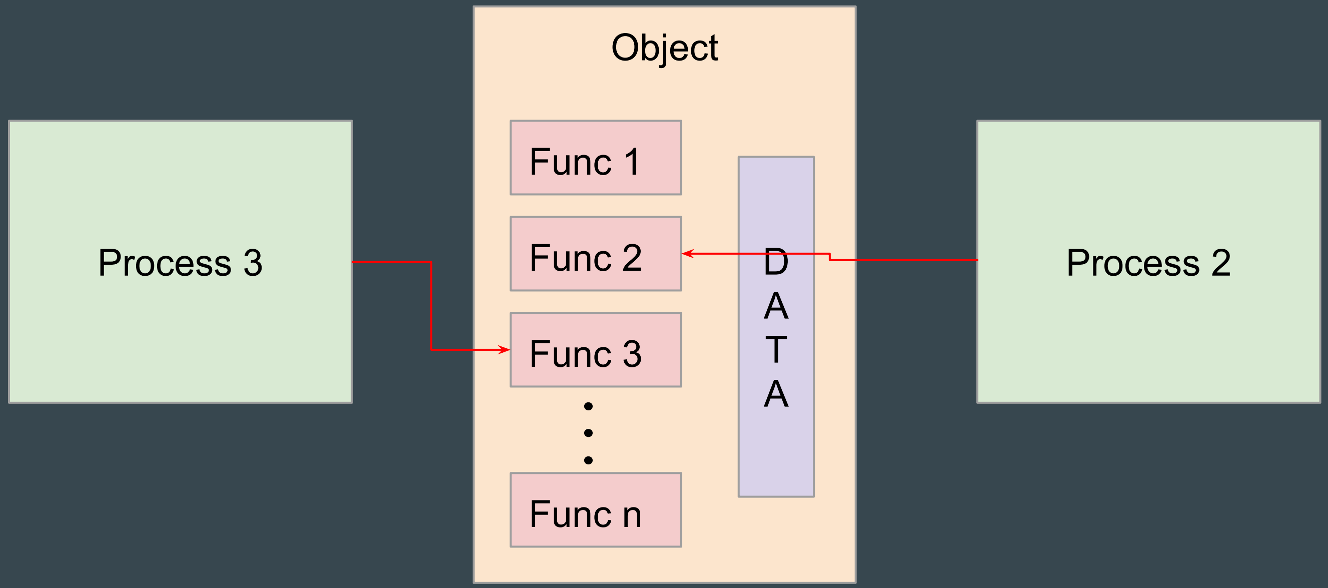
High Level View



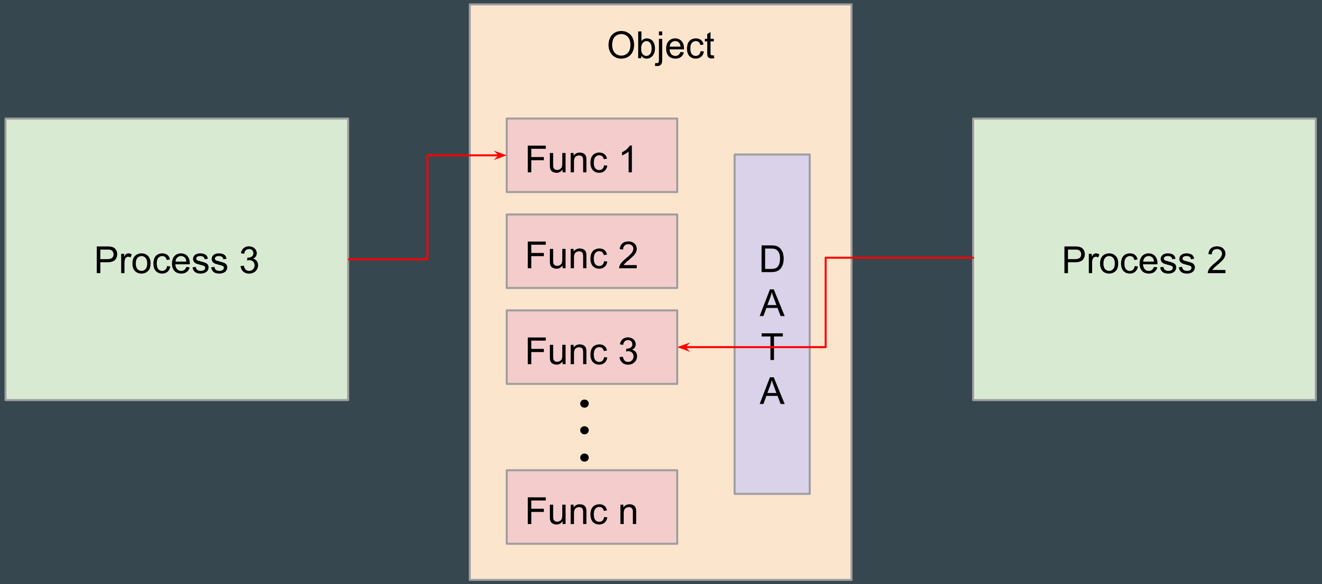
High Level View



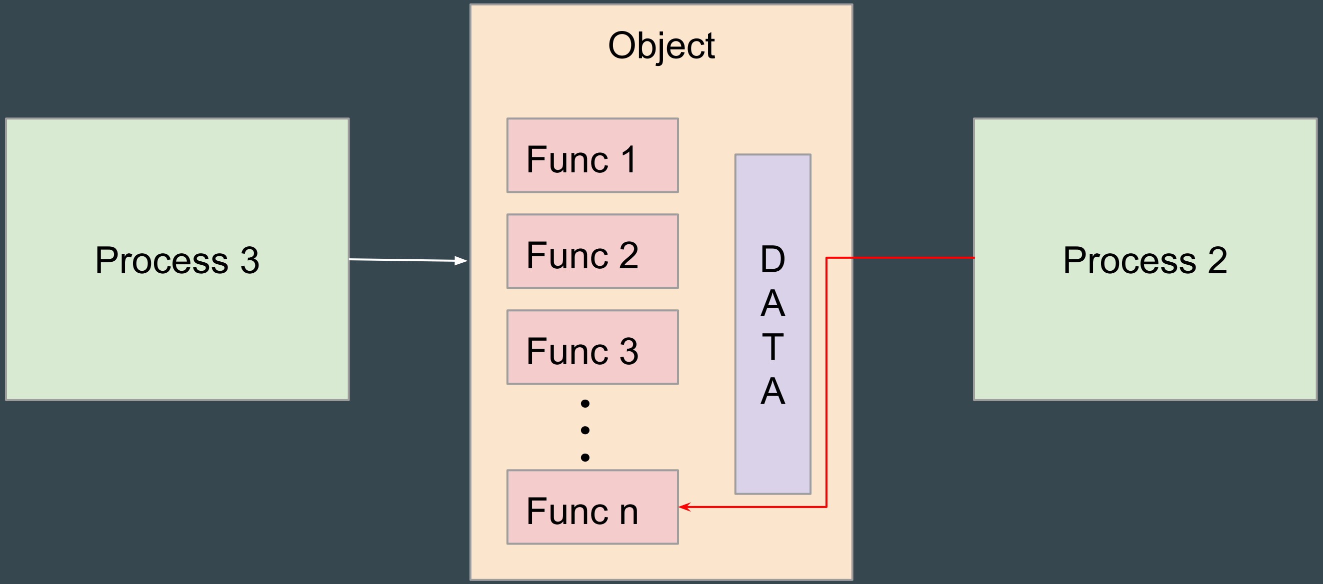
High Level View



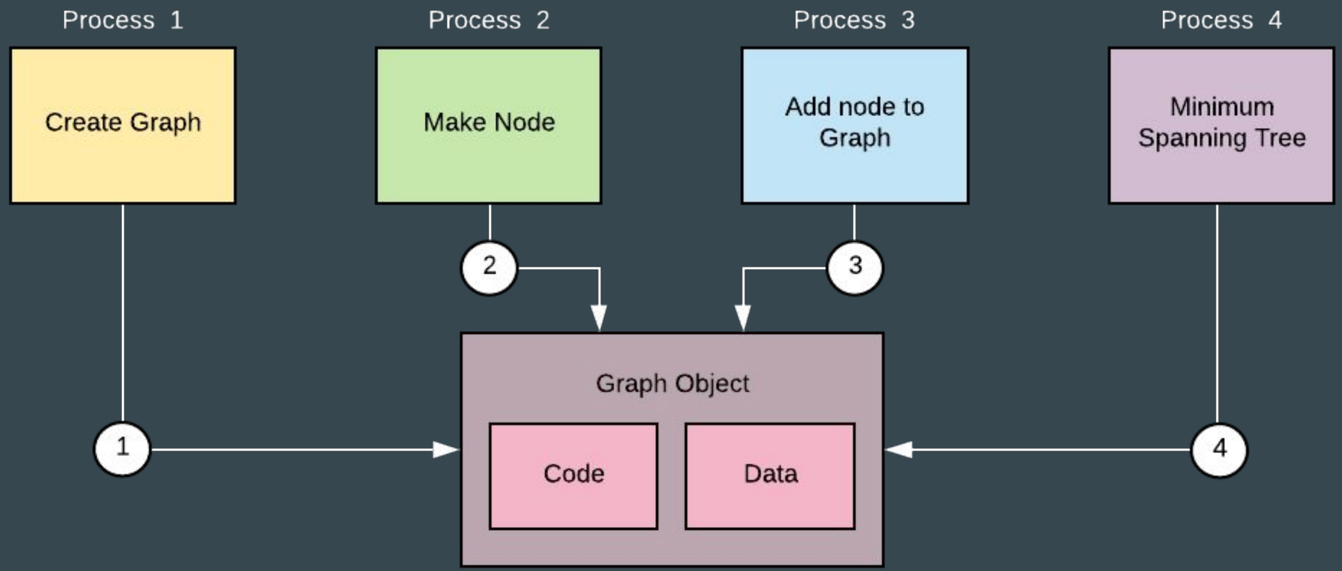
High Level View



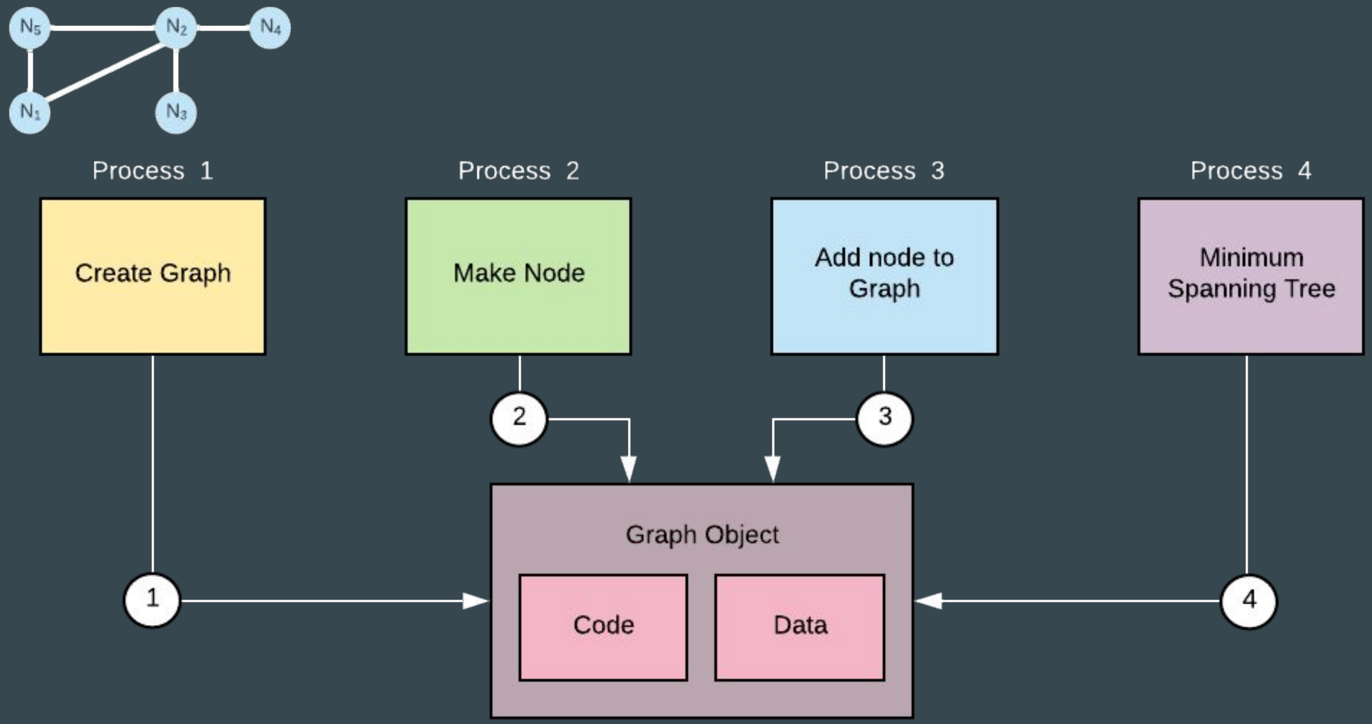
High Level View



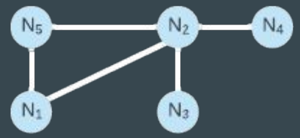
Example: Graph Processing



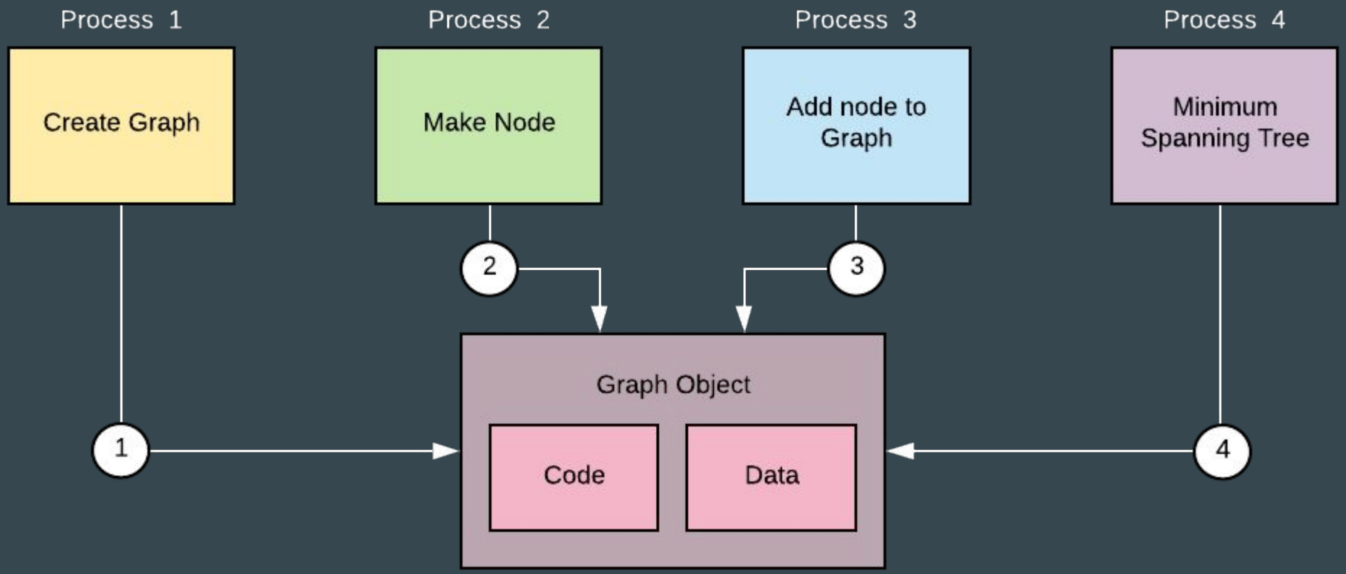
Example: Graph Processing



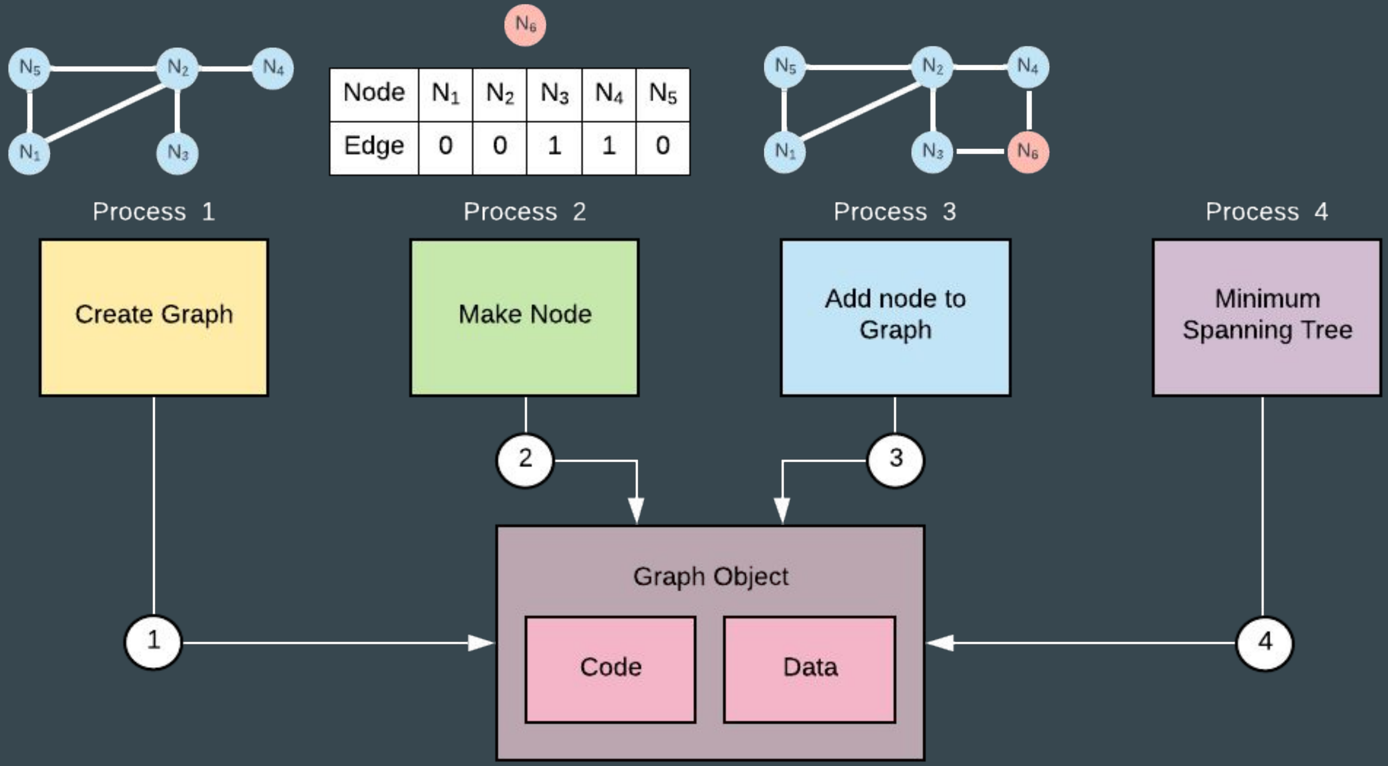
Example: Graph Processing



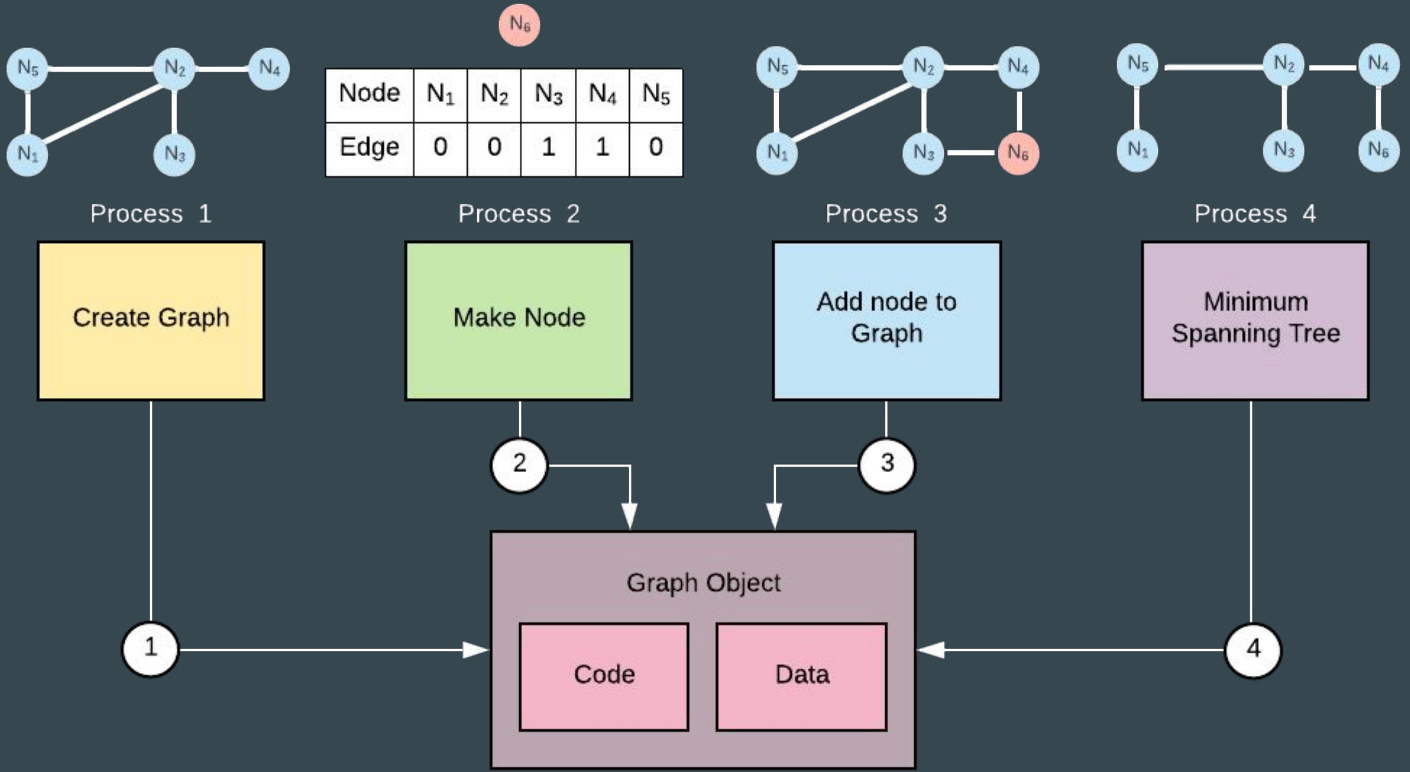
Node	N_1	N_2	N_3	N_4	N_5
Edge	0	0	1	1	0



Example: Graph Processing



Example: Graph Processing



Interface

- A kernel module needs to be inserted
  - Communication through IOCTL

Interface

- A kernel module needs to be inserted
  - Communication through IOCTL
- A user library, which needs to be linked to process

Interface

- A kernel module needs to be inserted
  - Communication through IOCTL
- A user library, which needs to be linked to process
  - Responsible for making all the calls related to the object framework

Interface

- A kernel module needs to be inserted
  - Communication through IOCTL
- A user library, which needs to be linked to process
  - Responsible for making all the calls related to the object framework
  - Obtains metadata of objects and carry out the required functionality

Interface: Library calls

- Init_object (binary):
  - Creates an object with the input binary

Interface: Library calls

- Init_object (binary):
  - Creates an object with the input binary
- Attach_object (obj_id):
  - Attaches the object in process' address space

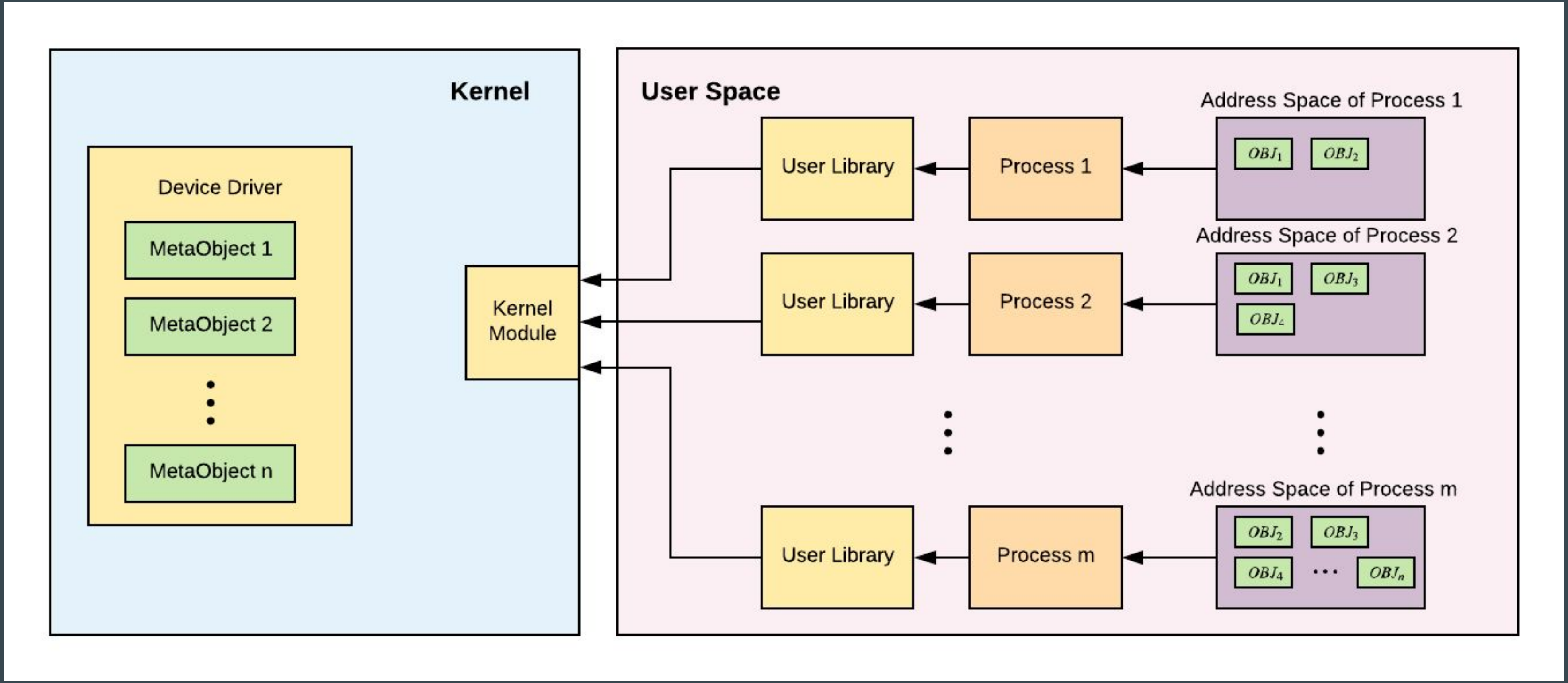
Interface: Library calls

- Init_object (binary):
  - Creates an object with the input binary
- Attach_object (obj_id):
  - Attaches the object in process' address space
- Call_func (obj_id, func_id):
  - Calls given function from given object

Interface: Library calls

- Init_object (binary):
  - Creates an object with the input binary
- Attach_object (obj_id):
  - Attaches the object in process' address space
- Call_func (obj_id, func_id):
  - Calls given function from given object
- Append_code (obj_id, binary):
  - Adds the given code in object's memory segment

Implementation: Framework



Implementation: Shared Memory

- Similar to implementation of Exec
  - Map code and data in address space`

Implementation: Shared Memory

- Similar to implementation of Exec
    - Map code and data in address space
    - But we only require a small subset of features

Implementation: Shared Memory

- Similar to implementation of Exec
  - Map code and data in address space
  - But we only require a small subset of features
  - High overhead, if already implemented functions are used

Implementation: Shared Memory

- Similar to implementation of Exec
  - Map code and data in address space
  - But we only require a small subset of features
  - High overhead, if already implemented functions are used
- Use Shared Memory Techniques
  - Shm (shmget, shmat, shmdt, shmctl)
  - Mmap

Implementation: Relocations

- Binary (code segment) of object can contain unresolved links (since binary is compiled code)
  - Functions in different library
  - Global Variables

Implementation: Relocations

- Binary (code segment) of object can contain unresolved links (since binary is compiled code)
  - Functions in different library
  - Global Variables
- Resolved using elf library
  - Relocation section contains relevant information
  - Target address can be found in symbol table of linked libraries

Applications

Object model is a basic framework which can be used in variety of fields

- Operating System for NVRAM
- Lambda Services
- Distributed Systems

Applications: Operating System

- Traditional process oriented OS mechanisms don’t have defined behaviour when stopped suddenly

Applications: Operating System

- Traditional process oriented OS mechanisms don't have defined behaviour when stopped suddenly
- Objects can be handled in defined manner

Applications: Operating System

- Traditional process oriented OS mechanisms don't have defined behaviour when stopped suddenly
- Objects can be handled in defined manner
- Replace process with Objects

Applications: Operating System

- Traditional process oriented OS mechanisms don’t have defined behaviour when stopped suddenly
- Objects can be handled in defined manner
- Replace process with Objects
- Make kernel itself an object
    - Requires composition of objects

Applications: Lambda Services

- Modern serverless online computing

Applications: Lambda Services

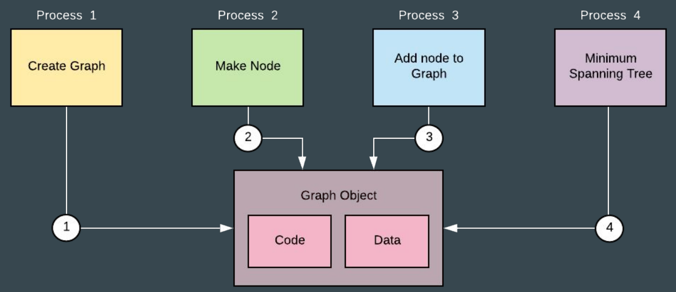
- Modern serverless online computing
- Takes code and data as input and outputs processed data using the code

Applications: Lambda Services

- Modern serverless online computing
- Takes code and data as input and outputs processed data using the code
- Object model fits in perfectly
  - Also provides stateful experience by keeping updated data in the object
  - Requires data input only once

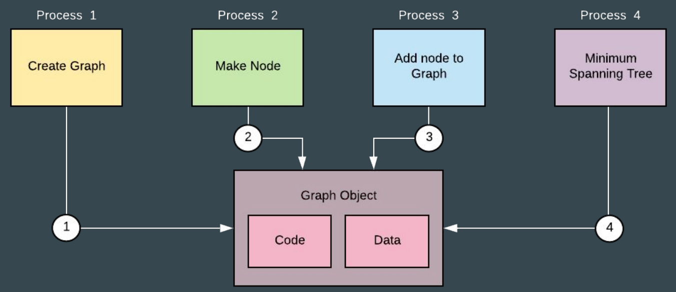
Applications: Distributed Systems

- Make objects the computing entities



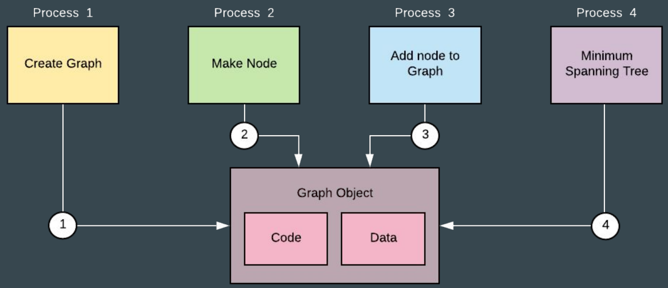
Applications: Distributed Systems

- Make objects the computing entities
- Takes the form of Pipeline computing



Applications: Distributed Systems

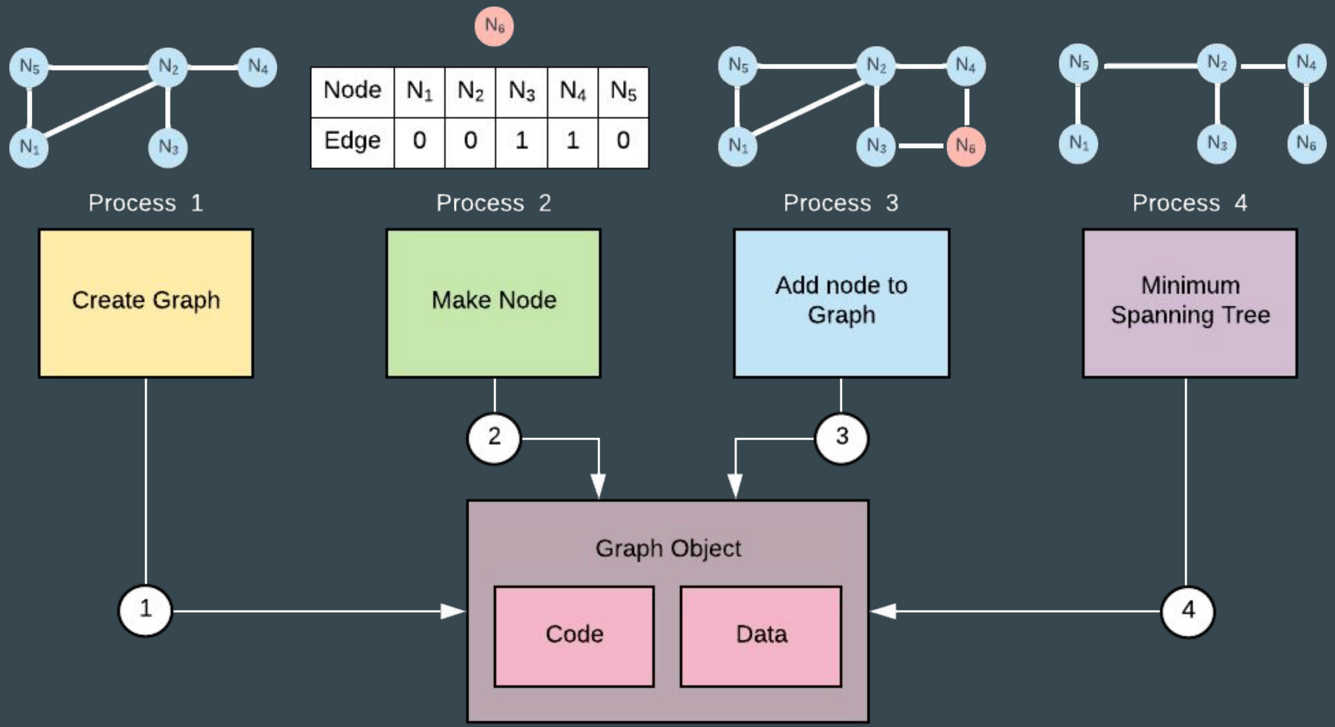
- Make objects the computing entities
- Takes the form of Pipeline computing
- Signals would automate the computing as well



Evaluation

- Basic Test Cases
  - Consistency Checks
  - Data Integrity
  - Function Calls
  - Add Code (binary)
  - Relocation checks
- Graph Processing

Evaluation: Graph Processing



Future Work

- Composition of objects

Future Work

- Composition of objects
- Signals from objects to objects and processes

Future Work

- Composition of objects
- Signals from objects to objects and processes
- Security Aspects
  - Since the object shares the space address space as process, isolation can be compromised

Future Work

- Composition of objects
- Signals from objects to objects and processes
- Security Aspects
  - Since the object shares the space address space as process, isolation can be compromised
- Integrate our object model in different applications

Conclusion

- NVRAMs need reforms in existing OS mechanisms

Conclusion

- NVRAMs need reforms in existing OS mechanisms
- Implemented a computational framework involving objects

Conclusion

- NVRAMs need reforms in existing OS mechanisms
- Implemented a computational framework involving objects
- Object model finds a lot of applications in modern day computing

Conclusion

- NVRAMs need reforms in existing OS mechanisms
- Implemented a computational framework involving objects
- Object model finds a lot of applications in modern day computing
- This is just a glimpse of the bigger picture that remains to come in future

Thank You !!

Back Up: NVM Characteristics

	SRAM	DRAM	HDD	NAND flash	STT-RAM	ReRAM	PCM	FeRAM
Cell size (F^2)	120–200	60–100	N/A	4–6	6–50	4–10	4–12	6–40
Write Endurance	10^{16}	$>10^{15}$	$>10^{15}$ (pb: mechanical parts)	10^4–10^5	10^{12}–10^{15}	10^8–10^{11}	10^8–10^9	10^{14}–10^{15}
Read Latency	~0.2–2ns	~10ns	3–5ms	15–35μs	2–35ns	~10ns	20–60ns	20–80ns
Write Latency	~0.2–2ns	~10ns	3–5ms	200–500μs	3–50ns	~50ns	20–150ns	50–75ns
Leakage Power	High	Medium	(Mechanical parts)	Low	Low	Low	Low	Low
Dynamic Energy (R/W)	Low	Medium	(Mechanical parts)	Low	Low/High	Low/High	Medium/ High	Low/High
Maturity	Mature	Mature	Mature	Mature	Test chips	Test chips	Test chips	Manufactured