

Athena

0.1

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	athena::backend::AbstractDevice Class Reference . . . . .	5
3.2	athena::backend::AbstractExecutor Class Reference . . . . .	6
3.2.1	Detailed Description . . . . .	6
3.2.2	Member Function Documentation . . . . .	6
3.2.2.1	execute() . . . . .	6
3.2.2.2	getMemoryManager() . . . . .	7
3.2.2.3	setBytecode() . . . . .	7
3.3	athena::core::loss::AbstractLossFunction Class Reference . . . . .	7
3.4	athena::backend::AbstractMemoryManager Class Reference . . . . .	8
3.4.1	Detailed Description . . . . .	8
3.4.2	Member Function Documentation . . . . .	8
3.4.2.1	addTensor() . . . . .	8
3.4.2.2	deleteFromMem() . . . . .	9
3.4.2.3	getPhysicalAddress() . . . . .	9
3.4.2.4	load() . . . . .	9
3.4.2.5	resetTable() . . . . .	10
3.4.2.6	unlock() . . . . .	10

3.5	<a href="#">athena::core::optimizers::AbstractOptimizer Class Reference</a>	10
3.6	<a href="#">athena::core::kernels::AddOpKernel Class Reference</a>	11
3.6.1	Detailed Description	11
3.6.2	Member Function Documentation	11
3.6.2.1	<a href="#">getDerivativeBytecode()</a>	11
3.6.2.2	<a href="#">getDerivativeShape()</a>	12
3.6.2.3	<a href="#">getOperandsCount()</a>	12
3.6.2.4	<a href="#">getOutputShape()</a>	12
3.7	<a href="#">athena::backend::generic::CPUDevice Class Reference</a>	13
3.7.1	Detailed Description	13
3.8	<a href="#">athena::backend::generic::GenericExecutor Class Reference</a>	14
3.8.1	Detailed Description	14
3.8.2	Member Function Documentation	14
3.8.2.1	<a href="#">execute()</a>	14
3.8.2.2	<a href="#">getMemoryManager()</a>	15
3.9	<a href="#">athena::backend::generic::GenericMemoryManager Class Reference</a>	15
3.9.1	Detailed Description	16
3.9.2	Member Function Documentation	16
3.9.2.1	<a href="#">deinit()</a>	16
3.9.2.2	<a href="#">deleteFromMem()</a>	16
3.9.2.3	<a href="#">getPhysicalAddress()</a>	16
3.9.2.4	<a href="#">init()</a>	17
3.9.2.5	<a href="#">load()</a> [1/2]	17
3.9.2.6	<a href="#">load()</a> [2/2]	17
3.9.2.7	<a href="#">processQueue()</a>	18
3.9.2.8	<a href="#">unlock()</a>	18
3.10	<a href="#">athena::core::optimizers::GradientDescent Class Reference</a>	18
3.11	<a href="#">athena::core::InputNode Class Reference</a>	19
3.11.1	Detailed Description	19
3.11.2	Member Function Documentation	19

3.11.2.1	<a href="#">after()</a>	20
3.11.2.2	<a href="#">getData()</a>	20
3.11.2.3	<a href="#">getMappedMemCell()</a>	20
3.11.2.4	<a href="#">isFrozen()</a>	20
3.11.2.5	<a href="#">isInputNode()</a>	21
3.11.2.6	<a href="#">setFrozen()</a>	21
3.11.2.7	<a href="#">setMappedMemCell()</a>	21
3.12	<a href="#">athena::core::kernels::MatMulOpKernel Class Reference</a>	21
3.12.1	<a href="#">Detailed Description</a>	22
3.12.2	<a href="#">Member Function Documentation</a>	22
3.12.2.1	<a href="#">getDerivativeBytecode()</a>	22
3.12.2.2	<a href="#">getDerivativeShape()</a>	23
3.12.2.3	<a href="#">getOperandsCount()</a>	23
3.12.2.4	<a href="#">getOutputShape()</a>	23
3.13	<a href="#">athena::backend::generic::MemoryChunk Struct Reference</a>	24
3.13.1	<a href="#">Detailed Description</a>	24
3.14	<a href="#">athena::core::loss::MSELoss Class Reference</a>	24
3.15	<a href="#">athena::core::loss::MSEOpKernel Class Reference</a>	25
3.15.1	<a href="#">Member Function Documentation</a>	25
3.15.1.1	<a href="#">getDerivativeBytecode()</a>	25
3.15.1.2	<a href="#">getDerivativeShape()</a>	26
3.15.1.3	<a href="#">getOperandsCount()</a>	26
3.15.1.4	<a href="#">getOutputShape()</a>	26
3.16	<a href="#">athena::core::Node Class Reference</a>	27
3.16.1	<a href="#">Detailed Description</a>	28
3.16.2	<a href="#">Member Function Documentation</a>	28
3.16.2.1	<a href="#">after()</a>	28
3.16.2.2	<a href="#">isInputNode()</a>	28
3.17	<a href="#">athena::core::OpKernel Class Reference</a>	29
3.17.1	<a href="#">Detailed Description</a>	29

3.17.2	Member Function Documentation	29
3.17.2.1	getDerivativeBytecode()	29
3.17.2.2	getDerivativeShape()	30
3.17.2.3	getOperandsCount()	30
3.17.2.4	getOutputShape()	30
3.18	athena::backend::generic::QueueItem Struct Reference	31
3.18.1	Detailed Description	31
3.19	athena::core::kernels::ScaleOpKernel Class Reference	31
3.19.1	Detailed Description	32
3.19.2	Member Function Documentation	32
3.19.2.1	getDerivativeBytecode()	32
3.19.2.2	getDerivativeShape()	32
3.19.2.3	getOperandsCount()	33
3.19.2.4	getOutputShape()	33
3.20	athena::core::Session Class Reference	34
3.20.1	Detailed Description	34
3.20.2	Member Function Documentation	34
3.20.2.1	prepare()	34
3.20.2.2	run()	34
3.21	athena::core::optimizers::SGDOptimizer Class Reference	35
3.22	athena::core::kernels::SigmoidOpKernel Class Reference	35
3.22.1	Detailed Description	36
3.22.2	Member Function Documentation	36
3.22.2.1	getDerivativeBytecode()	36
3.22.2.2	getDerivativeShape()	36
3.22.2.3	getOperandsCount()	37
3.22.2.4	getOutputShape()	37
3.23	athena::backend::generic::SwapRecord Struct Reference	37
3.23.1	Detailed Description	38
3.24	athena::core::Tensor Class Reference	38

3.24.1 Detailed Description . . . . .	38
3.25 athena::core::TensorShape Class Reference . . . . .	38
3.25.1 Detailed Description . . . . .	39
3.25.2 Member Function Documentation . . . . .	39
3.25.2.1 dim() . . . . .	39
3.25.2.2 dimensions() . . . . .	39
3.25.2.3 operator!=(()) . . . . .	40
3.25.2.4 operator==(()) . . . . .	41
3.25.2.5 totalSize() . . . . .	41
3.26 athena::backend::VirtualMemory Class Reference . . . . .	41
3.26.1 Detailed Description . . . . .	42
3.26.2 Member Function Documentation . . . . .	42
3.26.2.1 allocate() . . . . .	42
3.26.2.2 free() [1/2] . . . . .	42
3.26.2.3 free() [2/2] . . . . .	42
3.27 athena::backend::VMemoryBlock Struct Reference . . . . .	43
<b>Index</b>	<b>45</b>





# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

athena::backend::AbstractDevice . . . . .	5
athena::backend::generic::CPUDevice . . . . .	13
athena::backend::AbstractExecutor . . . . .	6
athena::backend::generic::GenericExecutor . . . . .	14
athena::backend::AbstractMemoryManager . . . . .	8
athena::backend::generic::GenericMemoryManager . . . . .	15
athena::core::optimizers::AbstractOptimizer . . . . .	10
athena::core::optimizers::GradientDescent . . . . .	18
athena::core::optimizers::SGDOptimizer . . . . .	35
athena::backend::generic::MemoryChunk . . . . .	24
athena::core::Node . . . . .	27
athena::core::InputNode . . . . .	19
athena::core::loss::AbstractLossFunction . . . . .	7
athena::core::loss::MSELoss . . . . .	24
athena::core::OpKernel . . . . .	29
athena::core::kernels::AddOpKernel . . . . .	11
athena::core::kernels::MatMulOpKernel . . . . .	21
athena::core::kernels::ScaleOpKernel . . . . .	31
athena::core::kernels::SigmoidOpKernel . . . . .	35
athena::core::loss::MSEOpKernel . . . . .	25
athena::backend::generic::QueueItem . . . . .	31
athena::core::Session . . . . .	34
athena::backend::generic::SwapRecord . . . . .	37
athena::core::Tensor . . . . .	38
athena::core::TensorShape . . . . .	38
athena::backend::VirtualMemory . . . . .	41
athena::backend::VMemoryBlock . . . . .	43



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">athena::backend::AbstractDevice</a>	5
<a href="#">athena::backend::AbstractExecutor</a>	6
<a href="#">athena::core::loss::AbstractLossFunction</a>	7
<a href="#">athena::backend::AbstractMemoryManager</a>	8
<a href="#">athena::core::optimizers::AbstractOptimizer</a>	10
<a href="#">athena::core::kernels::AddOpKernel</a>	11
<a href="#">athena::backend::generic::CPUDevice</a>	13
<a href="#">athena::backend::generic::GenericExecutor</a>	14
<a href="#">athena::backend::generic::GenericMemoryManager</a>	15
<a href="#">athena::core::optimizers::GradientDescent</a>	18
<a href="#">athena::core::InputNode</a>	19
<a href="#">athena::core::kernels::MatMulOpKernel</a>	21
<a href="#">athena::backend::generic::MemoryChunk</a>	24
<a href="#">athena::core::loss::MSELoss</a>	24
<a href="#">athena::core::loss::MSEOpKernel</a>	25
<a href="#">athena::core::Node</a>	27
<a href="#">athena::core::OpKernel</a>	29
<a href="#">athena::backend::generic::QueueItem</a>	31
<a href="#">athena::core::kernels::ScaleOpKernel</a>	31
<a href="#">athena::core::Session</a>	34
<a href="#">athena::core::optimizers::SGDOptimizer</a>	35
<a href="#">athena::core::kernels::SigmoidOpKernel</a>	35
<a href="#">athena::backend::generic::SwapRecord</a>	37
<a href="#">athena::core::Tensor</a>	38
<a href="#">athena::core::TensorShape</a>	38
<a href="#">athena::backend::VirtualMemory</a>	41
<a href="#">athena::backend::VMemoryBlock</a>	43

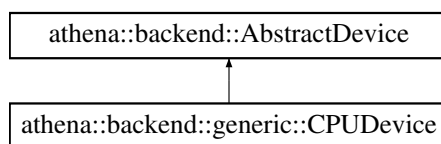


## Chapter 3

# Class Documentation

### 3.1 athena::backend::AbstractDevice Class Reference

Inheritance diagram for athena::backend::AbstractDevice:



#### Public Member Functions

- unsigned long **getMaxThreadMemSize** ()
- void **setMaxThreadMemSize** (unsigned long size=0)
- virtual [AbstractMemoryManager](#) \* **getMemoryManager** ()=0

#### Protected Attributes

- unsigned long **maxThreadMemorySize**
- unsigned long **maxThreads**
- unsigned long **memorySize**

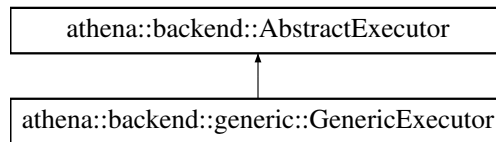
The documentation for this class was generated from the following files:

- backend/AbstractDevice.h
- backend/AbstractDevice.cpp

## 3.2 athena::backend::AbstractExecutor Class Reference

```
#include <AbstractExecutor.h>
```

Inheritance diagram for athena::backend::AbstractExecutor:



### Public Member Functions

- virtual void `execute` ()=0
- virtual `AbstractMemoryManager` \* `getMemoryManager` ()=0
- void `setBytecode` (std::vector< vm\_word > &bytecode)

### Protected Attributes

- std::vector< vm\_word > **bytecode**

### 3.2.1 Detailed Description

An Executor is a Virtual Machine that runs Athena `bytecode`. `AbstractExecutor` is the base class for all executors.

### 3.2.2 Member Function Documentation

#### 3.2.2.1 `execute()`

```
virtual void athena::backend::AbstractExecutor::execute ( ) [pure virtual]
```

Executes current bytecode. After execution threads state must be reset. However, memory state (Memory manager and its data) must persist.

Implemented in `athena::backend::generic::GenericExecutor`.

## 3.2.2.2 getMemoryManager()

```
virtual AbstractMemoryManager* athena::backend::AbstractExecutor::getMemoryManager ( ) [pure virtual]
```

## Returns

Memory Manager for current device

Implemented in [athena::backend::generic::GenericExecutor](#).

## 3.2.2.3 setBytecode()

```
void athena::backend::AbstractExecutor::setBytecode (
    std::vector< vm_word > & bytecode )
```

Sets new bytecode

## Parameters

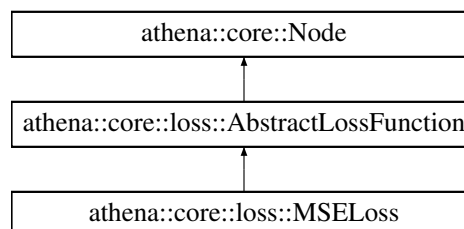
<i>bytecode</i>	Bytecode
-----------------	----------

The documentation for this class was generated from the following files:

- backend/AbstractExecutor.h
- backend/AbstractExecutor.cpp

## 3.3 athena::core::loss::AbstractLossFunction Class Reference

Inheritance diagram for athena::core::loss::AbstractLossFunction:



## Public Member Functions

- **AbstractLossFunction** ([OpKernel](#) \*)

## Additional Inherited Members

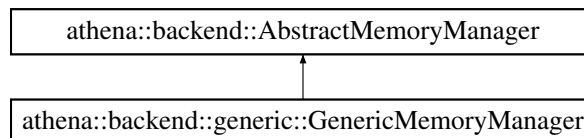
The documentation for this class was generated from the following files:

- `core/loss/AbstractLossFunction.h`
- `core/loss/AbstractLossFunction.cpp`

## 3.4 athena::backend::AbstractMemoryManager Class Reference

```
#include <AbstractMemoryManager.h>
```

Inheritance diagram for `athena::backend::AbstractMemoryManager`:



### Public Member Functions

- void `resetTable` ()
- void `addTensor` (`athena::core::Tensor` \*`tensor`)
- virtual void \* `getPhysicalAddress` (`vm_word` `virtualAddress`)=0
- void `load` (`athena::core::Tensor` \*`tensor`)
- void `load` (`vm_word` `address`)
- virtual void `load` (`vm_word` `address`, unsigned long `length`)=0
- virtual void `unlock` (`vm_word` `address`)=0
- virtual void `deleteFromMem` (`vm_word` `address`)=0
- `athena::core::Tensor` \* `getTensor` (`vm_word` `address`)

### Protected Attributes

- `std::list< athena::core::Tensor *>` `tensors`

### 3.4.1 Detailed Description

This class is an interface for physical memory managers. They provide conversion between virtual addresses and physical ones. A typical strategy for memory manager is to allocate as much memory as possible and then provide tensors with it. This class also encapsulates table of `athena::core::Tensor` objects. One can think of it as of variables table in a compiler.

### 3.4.2 Member Function Documentation

#### 3.4.2.1 addTensor()

```
void athena::backend::AbstractMemoryManager::addTensor (
    athena::core::Tensor * tensor )
```

Adds Tensor to table



## Parameters

<i>tensor</i>	Tensor, that will be added
---------------	----------------------------

## 3.4.2.2 deleteFromMem()

```
virtual void athena::backend::AbstractMemoryManager::deleteFromMem (
    vm_word address ) [pure virtual]
```

Mark corresponding memory chunk as free

## Parameters

<i>address</i>	Virtual address
----------------	-----------------

Implemented in [athena::backend::generic::GenericMemoryManager](#).

## 3.4.2.3 getPhysicalAddress()

```
virtual void* athena::backend::AbstractMemoryManager::getPhysicalAddress (
    vm_word virtualAddress ) [pure virtual]
```

Convert virtual address to physical one

## Parameters

<i>virtualAddress</i>	Virtual address, unsigned long from 0 to 2 <sup>64</sup> -1
-----------------------	---

## Returns

Pointer to physical memory

Implemented in [athena::backend::generic::GenericMemoryManager](#).

## 3.4.2.4 load()

```
virtual void athena::backend::AbstractMemoryManager::load (
    vm_word address,
    unsigned long length ) [pure virtual]
```

Move data to the fastest memory type available (e.g. from hard drive to RAM) and lock it (prevent from being offloaded)

## Parameters

<i>address</i>	Virtual address
<i>length</i>	Size of Tensor in bytes

Implemented in [athena::backend::generic::GenericMemoryManager](#).

### 3.4.2.5 resetTable()

```
void athena::backend::AbstractMemoryManager::resetTable ( )
```

Clears table of Tensors

### 3.4.2.6 unlock()

```
virtual void athena::backend::AbstractMemoryManager::unlock (
    vm_word address ) [pure virtual]
```

Lets data be offloaded to a slower memory type (e.g. from RAM to HDD)

## Parameters

<i>address</i>	Virtual address
----------------	-----------------

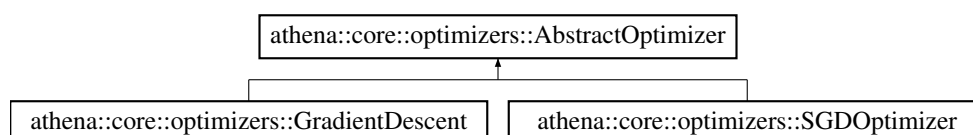
Implemented in [athena::backend::generic::GenericMemoryManager](#).

The documentation for this class was generated from the following files:

- backend/AbstractMemoryManager.h
- backend/AbstractMemoryManager.cpp

## 3.5 athena::core::optimizers::AbstractOptimizer Class Reference

Inheritance diagram for athena::core::optimizers::AbstractOptimizer:



### Public Member Functions

- **AbstractOptimizer** ([athena::core::loss::AbstractLossFunction](#) \*loss)
- void **init** ([Session](#) \*session)
- virtual void **prepare** ()=0
- virtual void **minimize** ()=0

### Protected Attributes

- `std::vector< InputNode *>` **headNodes**
- `std::vector< vm_word >` **bytecode**
- unsigned long **lastResultCell**
- [Session](#) \* **session**
- [athena::core::loss::AbstractLossFunction](#) \* **loss**

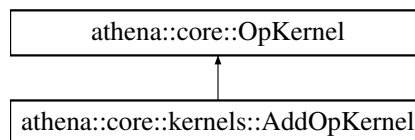
The documentation for this class was generated from the following files:

- `core/optimizers/AbstractOptimizer.h`
- `core/optimizers/AbstractOptimizer.cpp`

## 3.6 athena::core::kernels::AddOpKernel Class Reference

```
#include <AddOpKernel.h>
```

Inheritance diagram for `athena::core::kernels::AddOpKernel`:



### Public Member Functions

- `int` [getOperandsCount](#) () override
- [athena::core::TensorShape](#) & [getOutputShape](#) (std::vector< [athena::core::TensorShape](#) > &shapes) override
- [athena::core::TensorShape](#) & [getDerivativeShape](#) (int d, std::vector< [athena::core::TensorShape](#) > &shapes) override
- `std::vector< vm_word >` [getOpBytecode](#) (std::vector< vm\_word > args, vm\_word resultCell) override
- `std::vector< vm_word >` [getDerivativeBytecode](#) (int d, std::vector< vm\_word > args, vm\_word resultCell) override

### Additional Inherited Members

#### 3.6.1 Detailed Description

Performs sum of 2 given Tensors

#### 3.6.2 Member Function Documentation

##### 3.6.2.1 [getDerivativeBytecode\(\)](#)

```
std::vector< unsigned long > athena::core::kernels::AddOpKernel::getDerivativeBytecode (
    int d,
    std::vector< vm_word > args,
    vm_word resultCell ) [override], [virtual]
```

Generates bytecode to calculate partial derivative

## Parameters

<i>d</i>	Number of variable with respect to which derivative is calculated
<i>args</i>	Function arguments
<i>resultCell</i>	Number of memory cell where results are saved

## Returns

Implements [athena::core::OpKernel](#).

## 3.6.2.2 getDerivativeShape()

```
athena::core::TensorShape & athena::core::kernels::AddOpKernel::getDerivativeShape (
    int d,
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands

## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implements [athena::core::OpKernel](#).

## 3.6.2.3 getOperandsCount()

```
int athena::core::kernels::AddOpKernel::getOperandsCount ( ) [override], [virtual]
```

There can be unary, binary and other operations

## Returns

Number of operands accepted

Implements [athena::core::OpKernel](#).

## 3.6.2.4 getOutputShape()

```
athena::core::TensorShape & athena::core::kernels::AddOpKernel::getOutputShape (
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands

## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implements [athena::core::OpKernel](#).

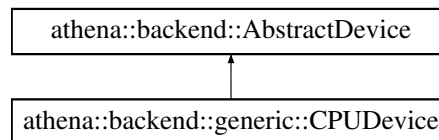
The documentation for this class was generated from the following files:

- core/kernels/AddOpKernel.h
- core/kernels/AddOpKernel.cpp

### 3.7 athena::backend::generic::CPUDevice Class Reference

```
#include <CPUDevice.h>
```

Inheritance diagram for athena::backend::generic::CPUDevice:



#### Public Member Functions

- [AbstractMemoryManager](#) \* **getMemoryManager** () override

#### Additional Inherited Members

#### 3.7.1 Detailed Description

This class represents a CPU It encapsulates Memory Manager

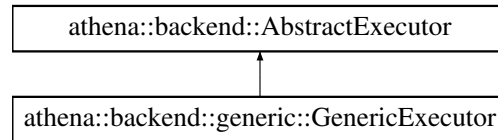
The documentation for this class was generated from the following files:

- backend/generic/CPUDevice.h
- backend/generic/CPUDevice.cpp

### 3.8 athena::backend::generic::GenericExecutor Class Reference

```
#include <GenericExecutor.h>
```

Inheritance diagram for athena::backend::generic::GenericExecutor:



#### Public Member Functions

- **GenericExecutor** ([CPUDevice](#) \*cpuDevice)
- void [execute](#) () override
- [AbstractMemoryManager](#) \* [getMemoryManager](#) () override

#### Additional Inherited Members

#### 3.8.1 Detailed Description

[GenericExecutor](#) is the state of the art implementation of [AbstractExecutor](#). While we try to make it work fast, the main goal of this implementation is to be mathematically correct and provide an example for more specific implementation.

[GenericExecutor](#) executes [bytecode](#) with standard CPU device. The actual implementations of bytecode commands use BLAS to speed up calculations. There are several accelerators available:

- [Apple Accelerate Framework](#)
- [OpenBLAS](#)
- [BLIS](#)

You can configure them during compile time. Other accelerators may be added later.

#### 3.8.2 Member Function Documentation

##### 3.8.2.1 execute()

```
void athena::backend::generic::GenericExecutor::execute ( ) [override], [virtual]
```

Executes current bytecode. After execution threads state must be reset. However, memory state (Memory manager and its data) must persist.

Implements [athena::backend::AbstractExecutor](#).

## 3.8.2.2 getMemoryManager()

```
athena::backend::AbstractMemoryManager * athena::backend::generic::GenericExecutor::getMemoryManager ( ) [override], [virtual]
```

## Returns

Memory Manager for current device

Implements [athena::backend::AbstractExecutor](#).

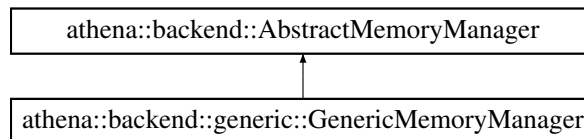
The documentation for this class was generated from the following files:

- backend/generic/GenericExecutor.h
- backend/generic/GenericExecutor.cpp

## 3.9 athena::backend::generic::GenericMemoryManager Class Reference

```
#include <GenericMemoryManager.h>
```

Inheritance diagram for athena::backend::generic::GenericMemoryManager:



## Public Member Functions

- void [init](#) ()
- void [deinit](#) ()
- void \* [getPhysicalAddress](#) (vm\_word virtualAddress) override
- void [load](#) (vm\_word address, unsigned long length) override
- void [unlock](#) (vm\_word address) override
- void [deleteFromMem](#) (vm\_word address) override
- void [setMemSize](#) (size\_t memSize)
- void [load](#) ([athena::core::Tensor](#) \*tensor)
- void [load](#) (vm\_word address)
- virtual void [load](#) (vm\_word address, unsigned long length)=0

## Protected Member Functions

- void [processQueue](#) (int laneId)

## Protected Attributes

- `std::list< SwapRecord *>` **swapRecords**
- `MemoryChunk *` **memoryChunksHead**
- `void *` **memory**
- `std::mutex` **memoryChunksLock**
- `std::vector< std::thread >` **memLanes**
- `size_t` **allocatedMemory**
- `std::queue< QueueItem *>` **loadQueue**
- `std::vector< bool >` **laneFinished**

### 3.9.1 Detailed Description

This class implements [AbstractMemoryManager](#) interface for [GenericExecutor](#). It pre-allocates RAM and uses persistent memory for swap. There are couple memory lanes - threads, that manage RAM. They monitor load↔Queue for new queries and move data from hard drive to RAM if needed.

### 3.9.2 Member Function Documentation

#### 3.9.2.1 deinit()

```
void athena::backend::generic::GenericMemoryManager::deinit ( )
```

Free RAM and stop all threads-memory lanes

#### 3.9.2.2 deleteFromMem()

```
void athena::backend::generic::GenericMemoryManager::deleteFromMem (
    vm_word address ) [override], [virtual]
```

Mark corresponding memory chunk as free

#### Parameters

<i>address</i>	Virtual address
----------------	-----------------

Implements [athena::backend::AbstractMemoryManager](#).

#### 3.9.2.3 getPhysicalAddress()

```
void * athena::backend::generic::GenericMemoryManager::getPhysicalAddress (
    vm_word virtualAddress ) [override], [virtual]
```

Convert virtual address to physical one



## Parameters

<i>virtualAddress</i>	Virtual address, unsigned long from 0 to $2^{64}-1$
-----------------------	---

## Returns

Pointer to physical memory

Implements [athena::backend::AbstractMemoryManager](#).

3.9.2.4 `init()`

```
void athena::backend::generic::GenericMemoryManager::init ( )
```

Initialize memory manager. That's where actual memory allocation happens. All configurations should be done before this method is called.

3.9.2.5 `load()` [1/2]

```
void athena::backend::generic::GenericMemoryManager::load (
    vm_word address,
    unsigned long length ) [override], [virtual]
```

Move data to the fastest memory type available (e.g. from hard drive to RAM) and lock it (prevent from being offloaded)

## Parameters

<i>address</i>	Virtual address
<i>length</i>	Size of Tensor in bytes

Implements [athena::backend::AbstractMemoryManager](#).

3.9.2.6 `load()` [2/2]

```
virtual void athena::backend::AbstractMemoryManager::load
```

Move data to the fastest memory type available (e.g. from hard drive to RAM) and lock it (prevent from being offloaded)

## Parameters

<i>address</i>	Virtual address
<i>length</i>	Size of Tensor in bytes

### 3.9.2.7 processQueue()

```
void athena::backend::generic::GenericMemoryManager::processQueue (
    int laneId )    [protected]
```

This is a thread function for memory lane-threads. It loads data to RAM and notifies corresponding threads

#### Parameters

<i>lane</i> ↔	
<i>Id</i>	

### 3.9.2.8 unlock()

```
void athena::backend::generic::GenericMemoryManager::unlock (
    vm_word address )    [override], [virtual]
```

Lets data be offloaded to a slower memory type (e.g. from RAM to HDD)

#### Parameters

<i>address</i>	Virtual address
----------------	-----------------

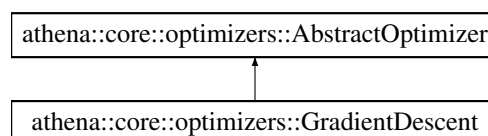
Implements [athena::backend::AbstractMemoryManager](#).

The documentation for this class was generated from the following files:

- backend/generic/GenericMemoryManager.h
- backend/generic/GenericMemoryManager.cpp

## 3.10 athena::core::optimizers::GradientDescent Class Reference

Inheritance diagram for athena::core::optimizers::GradientDescent:



### Public Member Functions

- **GradientDescent** ([athena::core::loss::AbstractLossFunction](#) \*loss, float learningRate)
- void **prepare** () override
- void **minimize** () override

### Protected Member Functions

- `std::tuple< std::vector< unsigned long >, unsigned long >` **getByteCode** ([athena::core::loss::AbstractLossFunction](#) \*node)

### Protected Attributes

- float **learningRate**

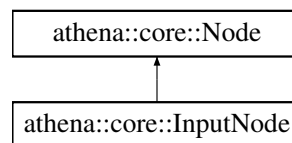
The documentation for this class was generated from the following files:

- `core/optimizers/GradientDescent.h`
- `core/optimizers/GradientDescent.cpp`

## 3.11 athena::core::InputNode Class Reference

```
#include <InputNode.h>
```

Inheritance diagram for `athena::core::InputNode`:



### Public Member Functions

- **InputNode** ([Tensor](#) \*input, bool [isFrozen](#)=true)
- bool [isInputNode](#) () override
- void [setMappedMemCell](#) (unsigned long cell)
- unsigned long [getMappedMemCell](#) ()
- void [after](#) ([Node](#) \*) override
- [Tensor](#) \* [getData](#) ()
- bool [isFrozen](#) ()
- void [setFrozen](#) (bool frozen)

### Additional Inherited Members

#### 3.11.1 Detailed Description

Subclass of [athena::core::Node](#) Represents a node that has no predecessors

#### 3.11.2 Member Function Documentation

### 3.11.2.1 after()

```
void athena::core::InputNode::after (
    Node * ) [inline], [override], [virtual]
```

InputNodes can't be placed after other nodes in Athena's execution graph. This method does nothing

Reimplemented from [athena::core::Node](#).

### 3.11.2.2 getData()

```
athena::core::Tensor * athena::core::InputNode::getData ( )
```

Get data associated with this [InputNode](#)

#### Returns

Pointer to [Tensor](#)

### 3.11.2.3 getMappedMemCell()

```
unsigned long athena::core::InputNode::getMappedMemCell ( )
```

Get the number of memory cell that is used to store tensor for this node

#### Returns

Memory cell number

### 3.11.2.4 isFrozen()

```
bool athena::core::InputNode::isFrozen ( )
```

InputNodes can be frozen. This means their tensors won't be changed during back propagation process (e.g. [InputNode](#) contains your input data). By default new InputNodes are frozen.

#### Returns

Current freeze state

## 3.11.2.5 isInputNode()

```
bool athena::core::InputNode::isInputNode ( ) [override], [virtual]
```

Check if it is an input node

## Returns

true

Reimplemented from [athena::core::Node](#).

## 3.11.2.6 setFrozen()

```
void athena::core::InputNode::setFrozen (
    bool frozen )
```

InputNodes can be frozen. This means their tensors won't be changed during back propagation process (e.g. [InputNode](#) contains your input data). By default new InputNodes are frozen.

## Parameters

<i>frozen</i>	True - freeze node, False - unfreeze node (make it variable)
---------------	--

## 3.11.2.7 setMappedMemCell()

```
void athena::core::InputNode::setMappedMemCell (
    unsigned long cell )
```

Specify which memory cell will be used to store tensor for this node

## Parameters

<i>cell</i>	Memory cell number
-------------	--------------------

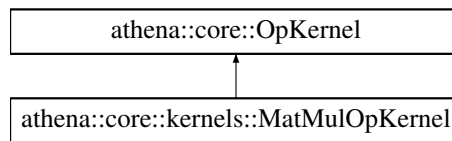
The documentation for this class was generated from the following files:

- core/InputNode.h
- core/InputNode.cpp

## 3.12 athena::core::kernels::MatMulOpKernel Class Reference

```
#include <MatMulOpKernel.h>
```

Inheritance diagram for athena::core::kernels::MatMulOpKernel:



## Public Member Functions

- int [getOperandsCount](#) () override
- [athena::core::TensorShape](#) & [getOutputShape](#) (std::vector< [athena::core::TensorShape](#) > &shapes) override
- [athena::core::TensorShape](#) & [getDerivativeShape](#) (int d, std::vector< [athena::core::TensorShape](#) > &shapes) override
- std::vector< vm\_word > [getOpBytecode](#) (std::vector< vm\_word > args, vm\_word resultCell) override
- std::vector< vm\_word > [getDerivativeBytecode](#) (int d, std::vector< vm\_word > args, vm\_word resultCell) override

## Additional Inherited Members

### 3.12.1 Detailed Description

Performs matrix multiplication of given Tensors. Matrix is a 2-D [Tensor](#). The main restriction for this operation is that the number of columns for the first column must be equal to the number of rows for the second matrix. The reason to introduce this operation apart from [Tensor](#) product is that it is widely adopted by different acceleration mechanism (BLAS, cuBLAS, Accelerate Framework, etc)

### 3.12.2 Member Function Documentation

#### 3.12.2.1 [getDerivativeBytecode\(\)](#)

```
std::vector< vm_word > athena::core::kernels::MatMulOpKernel::getDerivativeBytecode (
    int d,
    std::vector< vm_word > args,
    vm_word resultCell ) [override], [virtual]
```

Generates bytecode to calculate partial derivative

#### Parameters

<i>d</i>	Number of variable with respect to which derivative is calculated
<i>args</i>	Function arguments
<i>resultCell</i>	Number of memory cell where results are saved

## Returns

Implements [athena::core::OpKernel](#).

## 3.12.2.2 getDerivativeShape()

```
athena::core::TensorShape & athena::core::kernels::MatMulOpKernel::getDerivativeShape (
    int d,
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands

## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implements [athena::core::OpKernel](#).

## 3.12.2.3 getOperandsCount()

```
int athena::core::kernels::MatMulOpKernel::getOperandsCount ( ) [override], [virtual]
```

There can be unary, binary and other operations

## Returns

Number of operands accepted

Implements [athena::core::OpKernel](#).

## 3.12.2.4 getOutputShape()

```
athena::core::TensorShape & athena::core::kernels::MatMulOpKernel::getOutputShape (
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands

## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implements [athena::core::OpKernel](#).

The documentation for this class was generated from the following files:

- core/kernels/MatMulOpKernel.h
- core/kernels/MatMulOpKernel.cpp

### 3.13 athena::backend::generic::MemoryChunk Struct Reference

```
#include <GenericMemoryManager.h>
```

## Public Attributes

- `vm_word` **virtualAddress**
- `void *` **begin**
- `size_t` **length**
- `bool` **isFree**
- `bool` **isLocked**
- [MemoryChunk](#) \* **next**
- [MemoryChunk](#) \* **prev**

#### 3.13.1 Detailed Description

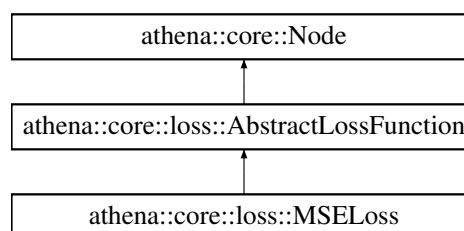
Describes single memory chunk that is allocated in RAM. Free status means there is no data in this chunk Locked status means this chunk is being used now and can't be unload to persistent memory.

The documentation for this struct was generated from the following file:

- backend/generic/GenericMemoryManager.h

### 3.14 athena::core::loss::MSELoss Class Reference

Inheritance diagram for athena::core::loss::MSELoss:





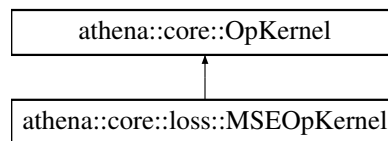
## Additional Inherited Members

The documentation for this class was generated from the following files:

- core/loss/MSELoss.h
- core/loss/MSELoss.cpp

## 3.15 athena::core::loss::MSEOpKernel Class Reference

Inheritance diagram for athena::core::loss::MSEOpKernel:



## Public Member Functions

- int [getOperandsCount](#) () override
- [athena::core::TensorShape](#) & [getOutputShape](#) (std::vector< [athena::core::TensorShape](#) > &shapes) override
- [athena::core::TensorShape](#) & [getDerivativeShape](#) (int, std::vector< [athena::core::TensorShape](#) > &shapes) override
- std::vector< unsigned long > [getOpBytecode](#) (std::vector< unsigned long > args, unsigned long resultCell) override
- std::vector< unsigned long > [getDerivativeBytecode](#) (int d, std::vector< unsigned long > args, unsigned long resultCell) override

## Additional Inherited Members

### 3.15.1 Member Function Documentation

#### 3.15.1.1 getDerivativeBytecode()

```
std::vector< unsigned long > athena::core::loss::MSEOpKernel::getDerivativeBytecode (
    int d,
    std::vector< unsigned long > args,
    unsigned long resultCell ) [override], [virtual]
```

Generates bytecode to calculate partial derivative

#### Parameters

<i>d</i>	Number of variable with respect to which derivative is calculated
<i>args</i>	Function arguments
<i>resultCell</i>	Number of memory cell where results are saved

## Returns

Implements [athena::core::OpKernel](#).

## 3.15.1.2 getDerivativeShape()

```
athena::core::TensorShape & athena::core::loss::MSEOpKernel::getDerivativeShape (
    int d,
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands

## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implements [athena::core::OpKernel](#).

## 3.15.1.3 getOperandsCount()

```
int athena::core::loss::MSEOpKernel::getOperandsCount ( ) [override], [virtual]
```

There can be unary, binary and other operations

## Returns

Number of operands accepted

Implements [athena::core::OpKernel](#).

## 3.15.1.4 getOutputShape()

```
athena::core::TensorShape & athena::core::loss::MSEOpKernel::getOutputShape (
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands

## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implements [athena::core::OpKernel](#).

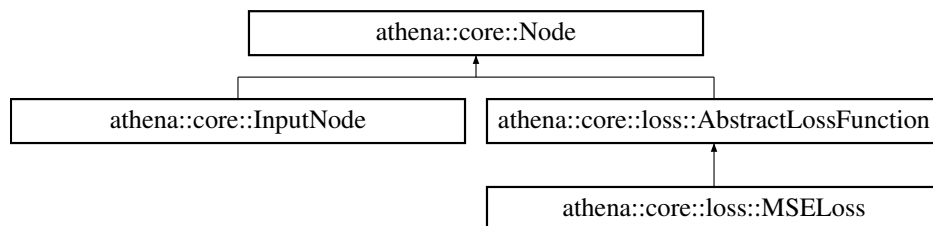
The documentation for this class was generated from the following files:

- core/loss/MSELoss.h
- core/loss/MSELoss.cpp

## 3.16 athena::core::Node Class Reference

```
#include <Node.h>
```

Inheritance diagram for athena::core::Node:



### Public Member Functions

- **Node** ([OpKernel](#) \*)
- virtual void **after** ([Node](#) \*predecessor)
- virtual bool **isInputNode** ()
- [OpKernel](#) \* **getOp** ()
- std::vector< [Node](#) \*> & **getIncomingNodes** ()
- std::string **getName** ()
- void **addDerivative** (unsigned long d)
- unsigned long **getDerivative** (int i)
- void **setCalculated** (unsigned long resCell)
- bool **isCalculated** ()
- unsigned long **getResult** ()
- void **updateUsageCount** ()
- bool **isGarbage** ()

### Protected Member Functions

- std::string **getRandomNodeName** ()

## Protected Attributes

- `std::vector< Node *>` **incomingNodes**
- `std::vector< Node *>` **outcomingNodes**
- `OpKernel *` **operation**
- `std::string` **name**
- `bool` **calculated**
- `std::vector< vm_word >` **derivatives**
- `unsigned long` **resultCell**
- `unsigned long` **usageCount**
- `bool` **derivativeMark**

### 3.16.1 Detailed Description

A basic element of execution graph Each node has pointers to its predecessors and successors. It encapsulates operation and data.

### 3.16.2 Member Function Documentation

#### 3.16.2.1 `after()`

```
void athena::core::Node::after (
    Node * predecessor ) [virtual]
```

Makes a new oriented edge in execution graph from predecessor to this node

#### Parameters

<i>predecessor</i>	A predecessor node
--------------------	--------------------

Reimplemented in [athena::core::InputNode](#).

#### 3.16.2.2 `isInputNode()`

```
bool athena::core::Node::isInputNode ( ) [virtual]
```

Check if it is an input node

#### Returns

false

Reimplemented in [athena::core::InputNode](#).

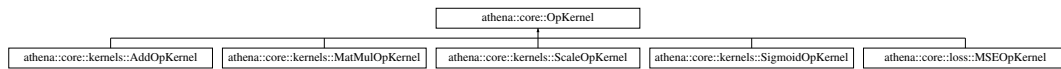
The documentation for this class was generated from the following files:

- `core/Node.h`
- `core/Node.cpp`

## 3.17 athena::core::OpKernel Class Reference

```
#include <OpKernel.h>
```

Inheritance diagram for athena::core::OpKernel:



### Public Member Functions

- **OpKernel** (OpCode opCode, std::string name)
- virtual int [getOperandsCount](#) ()=0
- virtual [athena::core::TensorShape](#) & [getOutputShape](#) (std::vector< [athena::core::TensorShape](#) > &shapes)=0
- virtual [athena::core::TensorShape](#) & [getDerivativeShape](#) (int d, std::vector< [athena::core::TensorShape](#) > &shapes)=0
- virtual std::vector< vm\_word > [getOpBytecode](#) (std::vector< vm\_word > args, vm\_word resultCell)=0
- virtual std::vector< vm\_word > [getDerivativeBytecode](#) (int d, std::vector< vm\_word > args, vm\_word resultCell)=0

### Protected Attributes

- OpCode **opCode**
- std::string **name**

#### 3.17.1 Detailed Description

Operation skeleton Each operation has OpCode

#### 3.17.2 Member Function Documentation

##### 3.17.2.1 [getDerivativeBytecode\(\)](#)

```
virtual std::vector< vm_word > athena::core::OpKernel::getDerivativeBytecode (
    int d,
    std::vector< vm_word > args,
    vm_word resultCell ) [pure virtual]
```

Generates bytecode to calculate partial derivative

#### Parameters

<i>d</i>	Number of variable with respect to which derivative is calculated
<i>args</i>	Function arguments
<i>resultCell</i>	Number of memory cell where results are saved

Generated by Doxygen

## Returns

Implemented in [athena::core::kernels::MatMulOpKernel](#), [athena::core::kernels::SigmoidOpKernel](#), [athena::core::loss::MSEOpKernel](#), [athena::core::kernels::AddOpKernel](#), and [athena::core::kernels::ScaleOpKernel](#).

## 3.17.2.2 getDerivativeShape()

```
virtual athena::core::TensorShape& athena::core::OpKernel::getDerivativeShape (
    int d,
    std::vector< athena::core::TensorShape > & shapes ) [pure virtual]
```

It is important for some operations to have certain size of their operands

## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implemented in [athena::core::kernels::MatMulOpKernel](#), [athena::core::kernels::SigmoidOpKernel](#), [athena::core::loss::MSEOpKernel](#), [athena::core::kernels::AddOpKernel](#), and [athena::core::kernels::ScaleOpKernel](#).

## 3.17.2.3 getOperandsCount()

```
virtual int athena::core::OpKernel::getOperandsCount ( ) [pure virtual]
```

There can be unary, binary and other operations

## Returns

Number of operands accepted

Implemented in [athena::core::kernels::MatMulOpKernel](#), [athena::core::kernels::SigmoidOpKernel](#), [athena::core::loss::MSEOpKernel](#), [athena::core::kernels::AddOpKernel](#), and [athena::core::kernels::ScaleOpKernel](#).

## 3.17.2.4 getOutputShape()

```
virtual athena::core::TensorShape& athena::core::OpKernel::getOutputShape (
    std::vector< athena::core::TensorShape > & shapes ) [pure virtual]
```

It is important for some operations to have certain size of their operands

## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implemented in [athena::core::kernels::MatMulOpKernel](#), [athena::core::kernels::SigmoidOpKernel](#), [athena::core::loss::MSEOpKernel](#), [athena::core::kernels::AddOpKernel](#), and [athena::core::kernels::ScaleOpKernel](#).

The documentation for this class was generated from the following file:

- `core/OpKernel.h`

## 3.18 athena::backend::generic::QueueItem Struct Reference

```
#include <GenericMemoryManager.h>
```

## Public Attributes

- `vm_word` **address**
- `size_t` **length**
- `bool` **alloc** = false
- `std::condition_variable` **loadHandle**
- `std::mutex` **m**
- `bool` **notified** = false

### 3.18.1 Detailed Description

Describes which Tensors should be loaded to RAM Alloc flag means we should not search for data in Swap

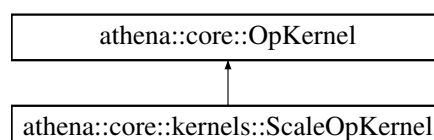
The documentation for this struct was generated from the following file:

- `backend/generic/GenericMemoryManager.h`

## 3.19 athena::core::kernels::ScaleOpKernel Class Reference

```
#include <ScaleOpKernel.h>
```

Inheritance diagram for `athena::core::kernels::ScaleOpKernel`:



## Public Member Functions

- `int getOperandsCount ()` override
- `athena::core::TensorShape & getOutputShape (std::vector< athena::core::TensorShape > &shapes)` override
- `athena::core::TensorShape & getDerivativeShape (int d, std::vector< athena::core::TensorShape > &shapes)` override
- `std::vector< vm_word > getOpBytecode (std::vector< vm_word > args, vm_word resultCell)` override
- `std::vector< vm_word > getDerivativeBytecode (int d, std::vector< vm_word > args, vm_word resultCell)` override

## Additional Inherited Members

### 3.19.1 Detailed Description

Multiply [Tensor](#) by scalar

### 3.19.2 Member Function Documentation

#### 3.19.2.1 getDerivativeBytecode()

```
std::vector< vm_word > athena::core::kernels::ScaleOpKernel::getDerivativeBytecode (
    int d,
    std::vector< vm_word > args,
    vm_word resultCell ) [override], [virtual]
```

Generates bytecode to calculate partial derivative

#### Parameters

<i>d</i>	Number of variable with respect to which derivative is calculated
<i>args</i>	Function arguments
<i>resultCell</i>	Number of memory cell where results are saved

#### Returns

Implements [athena::core::OpKernel](#).

#### 3.19.2.2 getDerivativeShape()

```
athena::core::TensorShape & athena::core::kernels::ScaleOpKernel::getDerivativeShape (
    int d,
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands



## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implements [athena::core::OpKernel](#).

## 3.19.2.3 getOperandsCount()

```
int athena::core::kernels::ScaleOpKernel::getOperandsCount ( ) [override], [virtual]
```

There can be unary, binary and other operations

## Returns

Number of operands accepted

Implements [athena::core::OpKernel](#).

## 3.19.2.4 getOutputShape()

```
athena::core::TensorShape & athena::core::kernels::ScaleOpKernel::getOutputShape (
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands

## Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

## Returns

New shape

Implements [athena::core::OpKernel](#).

The documentation for this class was generated from the following files:

- core/kernels/ScaleOpKernel.h
- core/kernels/ScaleOpKernel.cpp

## 3.20 athena::core::Session Class Reference

```
#include <Session.h>
```

### Public Member Functions

- void [prepare](#) ([Node](#) \*logits)
- [Tensor](#) \* [run](#) ()
- unsigned long [getResultCell](#) ()
- void [setExecutor](#) ([athena::backend::AbstractExecutor](#) \*exec)

### 3.20.1 Detailed Description

The class encapsulates everything needed for a single training step

### 3.20.2 Member Function Documentation

#### 3.20.2.1 prepare()

```
void athena::core::Session::prepare (
    Node * logits )
```

Generates bytecode for the whole graph

#### Parameters

<i>logits</i>	
---------------	--

#### 3.20.2.2 run()

```
athena::core::Tensor * athena::core::Session::run ( )
```

does single training step

#### Returns

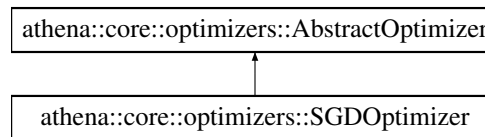
result tensor

The documentation for this class was generated from the following files:

- core/Session.h
- core/Session.cpp

## 3.21 athena::core::optimizers::SGDOptimizer Class Reference

Inheritance diagram for athena::core::optimizers::SGDOptimizer:



### Public Member Functions

- **SGDOptimizer** ([athena::core::loss::AbstractLossFunction](#) \*logits)

### Additional Inherited Members

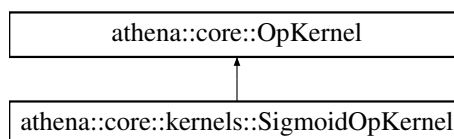
The documentation for this class was generated from the following file:

- core/optimizers/SGDOptimizer.h

## 3.22 athena::core::kernels::SigmoidOpKernel Class Reference

```
#include <SigmoidOpKernel.h>
```

Inheritance diagram for athena::core::kernels::SigmoidOpKernel:



### Public Member Functions

- int [getOperandsCount](#) () override
- [athena::core::TensorShape](#) & [getOutputShape](#) (std::vector< [athena::core::TensorShape](#) > &shapes) override
- [athena::core::TensorShape](#) & [getDerivativeShape](#) (int d, std::vector< [athena::core::TensorShape](#) > &shapes) override
- std::vector< vm\_word > [getOpBytecode](#) (std::vector< vm\_word > args, vm\_word resultCell) override
- std::vector< vm\_word > [getDerivativeBytecode](#) (int d, std::vector< vm\_word > args, vm\_word resultCell) override

## Additional Inherited Members

### 3.22.1 Detailed Description

Apply sigmoid function to every element of [Tensor](#). See [https://en.wikipedia.org/wiki/Sigmoid\\_function](https://en.wikipedia.org/wiki/Sigmoid_function) for more info

### 3.22.2 Member Function Documentation

#### 3.22.2.1 getDerivativeBytecode()

```
std::vector< vm_word > athena::core::kernels::SigmoidOpKernel::getDerivativeBytecode (
    int d,
    std::vector< vm_word > args,
    vm_word resultCell ) [override], [virtual]
```

Generates bytecode to calculate partial derivative

##### Parameters

<i>d</i>	Number of variable with respect to which derivative is calculated
<i>args</i>	Function arguments
<i>resultCell</i>	Number of memory cell where results are saved

##### Returns

Implements [athena::core::OpKernel](#).

#### 3.22.2.2 getDerivativeShape()

```
athena::core::TensorShape & athena::core::kernels::SigmoidOpKernel::getDerivativeShape (
    int d,
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands

##### Parameters

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

**Returns**

New shape

Implements [athena::core::OpKernel](#).

**3.22.2.3 getOperandsCount()**

```
int athena::core::kernels::SigmoidOpKernel::getOperandsCount ( ) [override], [virtual]
```

There can be unary, binary and other operations

**Returns**

Number of operands accepted

Implements [athena::core::OpKernel](#).

**3.22.2.4 getOutputShape()**

```
athena::core::TensorShape & athena::core::kernels::SigmoidOpKernel::getOutputShape (
    std::vector< athena::core::TensorShape > & shapes ) [override], [virtual]
```

It is important for some operations to have certain size of their operands

**Parameters**

<i>shape</i>	Original operand shape
<i>dim</i>	Dimensionality

**Returns**

New shape

Implements [athena::core::OpKernel](#).

The documentation for this class was generated from the following files:

- core/kernels/SigmoidOpKernel.h
- core/kernels/SigmoidOpKernel.cpp

**3.23 athena::backend::generic::SwapRecord Struct Reference**

```
#include <GenericMemoryManager.h>
```

## Public Attributes

- `vm_word` **address**
- `size_t` **length**
- `std::string` **filename**

### 3.23.1 Detailed Description

Describes single swap record - a file, that stores Tensor data

The documentation for this struct was generated from the following file:

- `backend/generic/GenericMemoryManager.h`

## 3.24 athena::core::Tensor Class Reference

```
#include <Tensor.h>
```

## Public Member Functions

- **Tensor** (const [TensorShape](#) &shape, `DataType` dataType)
- const [TensorShape](#) & **getShape** () const
- `DataType` **getType** () const
- `vm_word` **getStartAddress** ()
- void **setStartAddress** (`vm_word` address)
- [Tensor](#) & **operator[]** (unsigned int idx)

### 3.24.1 Detailed Description

In mathematics **tensor** is an abstract object, expressing some definite type of multi-linear concept. See [Wikipedia](#) for more info.

In Athena [Tensor](#) is an abstraction to represent data inside computational graph. A 1-dimensional [Tensor](#) is either scalar or vector. A 2-dimensional [Tensor](#) is a matrix.

The documentation for this class was generated from the following files:

- `core/Tensor.h`
- `core/Tensor.cpp`

## 3.25 athena::core::TensorShape Class Reference

```
#include <TensorShape.h>
```

## Public Member Functions

- **TensorShape** (std::vector< size\_t > shape)
- **TensorShape** (unsigned long \*shape, unsigned long length)
- **TensorShape** (const [TensorShape](#) &)
- **TensorShape & operator=** (const [TensorShape](#) &)
- unsigned long **dimensions** () const
- unsigned long **dim** (unsigned long n) const
- unsigned long **totalSize** () const
- const std::vector< unsigned long > & **getShape** () const
- bool **operator==** (const [TensorShape](#) &) const
- bool **operator!=** (const [TensorShape](#) &rhs) const

### 3.25.1 Detailed Description

Class represents size parameters for [Tensor](#)

### 3.25.2 Member Function Documentation

#### 3.25.2.1 dim()

```
unsigned long athena::core::TensorShape::dim (
    unsigned long n ) const
```

Gives size for certain dimension

##### Parameters

<i>n</i>	Dimension index ( 0 <= d < dimensions )
----------	---

##### Returns

Size of dimension n

#### 3.25.2.2 dimensions()

```
unsigned long athena::core::TensorShape::dimensions ( ) const
```

##### Returns

Number of dimensions in [Tensor](#)

### 3.25.2.3 operator"!=()

```
bool athena::core::TensorShape::operator!= (
    const TensorShape & rhs ) const
```



## Parameters

<i>rhs</i>	<a href="#">TensorShape</a> to be compared with
------------	---

## Returns

True if dimensions are different, else False

## 3.25.2.4 operator==()

```
bool athena::core::TensorShape::operator== (
    const TensorShape & rhs ) const
```

## Returns

True if dimensions are equal, else False

## 3.25.2.5 totalSize()

```
unsigned long athena::core::TensorShape::totalSize ( ) const
```

## Returns

Total number of elements in [Tensor](#)

The documentation for this class was generated from the following files:

- core/TensorShape.h
- core/TensorShape.cpp

## 3.26 athena::backend::VirtualMemory Class Reference

```
#include <VirtualMemory.h>
```

## Public Member Functions

- `vm_word` [allocate](#) ([athena::core::Tensor](#) \*tensor)
- void [free](#) ([athena::core::Tensor](#) \*tensor)
- void [free](#) (vm\_word virtualAddress)

### 3.26.1 Detailed Description

Virtual memory is an abstraction of storage resources that are actually available on a given machine. Each thread has its own address space. In Athena's VM address space is linear. This means that valid addresses are 0 to  $2^{64} - 1$ . Address 0 is reserved for NULL value. When Tensor is initialized, it is given with a continuous block of virtual addresses. When one actually needs to access Tensor's data, Memory Manager allocates physical memory and converts virtual addresses to physical ones. This helps Athena to run in low-memory conditions. This class is heavily used in Session class to generate bytecode.

To discover more about Virtual Memory see article on [Wikipedia](#)

### 3.26.2 Member Function Documentation

#### 3.26.2.1 allocate()

```
vm_word athena::backend::VirtualMemory::allocate (
    athena::core::Tensor * tensor )
```

Allocates virtual memory for given Tensor

##### Parameters

<i>tensor</i>	Tensor object
---------------	---------------

##### Returns

Virtual Address of 0 element of Tensor

#### 3.26.2.2 free() [1/2]

```
void athena::backend::VirtualMemory::free (
    athena::core::Tensor * tensor )
```

Marks memory as free

##### Parameters

<i>tensor</i>	Corresponding tensor
---------------	----------------------

#### 3.26.2.3 free() [2/2]

```
void athena::backend::VirtualMemory::free (
    vm_word virtualAddress )
```

Marks memory as free

#### Parameters

<i>virtualAddress</i>	
-----------------------	--

The documentation for this class was generated from the following files:

- backend/VirtualMemory.h
- backend/VirtualMemory.cpp

## 3.27 athena::backend::VMemoryBlock Struct Reference

### Public Attributes

- bool **isUsed**
- vm\_word **startAddress**
- vm\_word **endAddress**
- [VMemoryBlock](#) \* **nextBlock**
- [VMemoryBlock](#) \* **prevBlock**

The documentation for this struct was generated from the following file:

- backend/VirtualMemory.h



# Index

- addTensor
  - athena::backend::AbstractMemoryManager, 8
- after
  - athena::core::InputNode, 19
  - athena::core::Node, 28
- allocate
  - athena::backend::VirtualMemory, 42
- athena::backend::AbstractDevice, 5
- athena::backend::AbstractExecutor, 6
  - execute, 6
  - getMemoryManager, 6
  - setBytecode, 7
- athena::backend::AbstractMemoryManager, 8
  - addTensor, 8
  - deleteFromMem, 9
  - getPhysicalAddress, 9
  - load, 9
  - resetTable, 10
  - unlock, 10
- athena::backend::VMemoryBlock, 43
- athena::backend::VirtualMemory, 41
  - allocate, 42
  - free, 42
- athena::backend::generic::CPUDevice, 13
- athena::backend::generic::GenericExecutor, 14
  - execute, 14
  - getMemoryManager, 14
- athena::backend::generic::GenericMemoryManager, 15
  - deinit, 16
  - deleteFromMem, 16
  - getPhysicalAddress, 16
  - init, 17
  - load, 17
  - processQueue, 18
  - unlock, 18
- athena::backend::generic::MemoryChunk, 24
- athena::backend::generic::QueueItem, 31
- athena::backend::generic::SwapRecord, 37
- athena::core::InputNode, 19
  - after, 19
  - getData, 20
  - getMappedMemCell, 20
  - isFrozen, 20
  - isInputNode, 20
  - setFrozen, 21
  - setMappedMemCell, 21
- athena::core::Node, 27
  - after, 28
  - isInputNode, 28
- athena::core::OpKernel, 29
  - getDerivativeBytecode, 29
  - getDerivativeShape, 30
  - getOperandsCount, 30
  - getOutputShape, 30
- athena::core::Session, 34
  - prepare, 34
  - run, 34
- athena::core::Tensor, 38
- athena::core::TensorShape, 38
  - dim, 39
  - dimensions, 39
  - operator!=, 39
  - operator==, 41
  - totalSize, 41
- athena::core::kernels::AddOpKernel, 11
  - getDerivativeBytecode, 11
  - getDerivativeShape, 12
  - getOperandsCount, 12
  - getOutputShape, 12
- athena::core::kernels::MatMulOpKernel, 21
  - getDerivativeBytecode, 22
  - getDerivativeShape, 23
  - getOperandsCount, 23
  - getOutputShape, 23
- athena::core::kernels::ScaleOpKernel, 31
  - getDerivativeBytecode, 32
  - getDerivativeShape, 32
  - getOperandsCount, 33
  - getOutputShape, 33
- athena::core::kernels::SigmoidOpKernel, 35
  - getDerivativeBytecode, 36
  - getDerivativeShape, 36
  - getOperandsCount, 37
  - getOutputShape, 37
- athena::core::loss::AbstractLossFunction, 7
- athena::core::loss::MSELoss, 24
- athena::core::loss::MSEOpKernel, 25
  - getDerivativeBytecode, 25
  - getDerivativeShape, 26
  - getOperandsCount, 26
  - getOutputShape, 26
- athena::core::optimizers::AbstractOptimizer, 10
- athena::core::optimizers::GradientDescent, 18
- athena::core::optimizers::SGDOptimizer, 35
- deinit
  - athena::backend::generic::GenericMemoryManager, 16
- deleteFromMem

- athena::backend::AbstractMemoryManager, 9
- athena::backend::generic::GenericMemory↔  
Manager, 16
- dim
  - athena::core::TensorShape, 39
- dimensions
  - athena::core::TensorShape, 39
- execute
  - athena::backend::AbstractExecutor, 6
  - athena::backend::generic::GenericExecutor, 14
- free
  - athena::backend::VirtualMemory, 42
- getData
  - athena::core::InputNode, 20
- getDerivativeBytecode
  - athena::core::OpKernel, 29
  - athena::core::kernels::AddOpKernel, 11
  - athena::core::kernels::MatMulOpKernel, 22
  - athena::core::kernels::ScaleOpKernel, 32
  - athena::core::kernels::SigmoidOpKernel, 36
  - athena::core::loss::MSEOpKernel, 25
- getDerivativeShape
  - athena::core::OpKernel, 30
  - athena::core::kernels::AddOpKernel, 12
  - athena::core::kernels::MatMulOpKernel, 23
  - athena::core::kernels::ScaleOpKernel, 32
  - athena::core::kernels::SigmoidOpKernel, 36
  - athena::core::loss::MSEOpKernel, 26
- getMappedMemCell
  - athena::core::InputNode, 20
- getMemoryManager
  - athena::backend::AbstractExecutor, 6
  - athena::backend::generic::GenericExecutor, 14
- getOperandsCount
  - athena::core::OpKernel, 30
  - athena::core::kernels::AddOpKernel, 12
  - athena::core::kernels::MatMulOpKernel, 23
  - athena::core::kernels::ScaleOpKernel, 33
  - athena::core::kernels::SigmoidOpKernel, 37
  - athena::core::loss::MSEOpKernel, 26
- getOutputShape
  - athena::core::OpKernel, 30
  - athena::core::kernels::AddOpKernel, 12
  - athena::core::kernels::MatMulOpKernel, 23
  - athena::core::kernels::ScaleOpKernel, 33
  - athena::core::kernels::SigmoidOpKernel, 37
  - athena::core::loss::MSEOpKernel, 26
- getPhysicalAddress
  - athena::backend::AbstractMemoryManager, 9
  - athena::backend::generic::GenericMemory↔  
Manager, 16
- init
  - athena::backend::generic::GenericMemory↔  
Manager, 17
- isFrozen
  - athena::core::InputNode, 20
- isInputNode
  - athena::core::InputNode, 20
  - athena::core::Node, 28
- load
  - athena::backend::AbstractMemoryManager, 9
  - athena::backend::generic::GenericMemory↔  
Manager, 17
- operator!=
  - athena::core::TensorShape, 39
- operator==
  - athena::core::TensorShape, 41
- prepare
  - athena::core::Session, 34
- processQueue
  - athena::backend::generic::GenericMemory↔  
Manager, 18
- resetTable
  - athena::backend::AbstractMemoryManager, 10
- run
  - athena::core::Session, 34
- setBytecode
  - athena::backend::AbstractExecutor, 7
- setFrozen
  - athena::core::InputNode, 21
- setMappedMemCell
  - athena::core::InputNode, 21
- totalSize
  - athena::core::TensorShape, 41
- unlock
  - athena::backend::AbstractMemoryManager, 10
  - athena::backend::generic::GenericMemory↔  
Manager, 18