

Athena

0.1

Generated by Doxygen

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	2
2.1	Class List	2
3	File Index	3
3.1	File List	3
4	Class Documentation	5
4.1	athena::backend::AbstractDevice Class Reference	5
4.1.1	Detailed Description	5
4.2	athena::backend::AbstractExecutor Class Reference	6
4.2.1	Detailed Description	6
4.2.2	Member Function Documentation	6
4.2.2.1	execute()	6
4.2.2.2	getMemoryManager()	7
4.2.2.3	setBytecode()	7
4.3	athena::core::initializers::AbstractInitializer Class Reference	7
4.3.1	Detailed Description	8
4.3.2	Member Function Documentation	8
4.3.2.1	initialize()	8
4.4	athena::core::loss::AbstractLossFunction Class Reference	8
4.4.1	Detailed Description	9
4.5	athena::backend::AbstractMemoryManager Class Reference	9
4.5.1	Detailed Description	10

4.5.2	Member Function Documentation	10
4.5.2.1	addTensor()	10
4.5.2.2	allocateAndLock() [1/3]	10
4.5.2.3	allocateAndLock() [2/3]	10
4.5.2.4	allocateAndLock() [3/3]	12
4.5.2.5	deleteFromMem()	12
4.5.2.6	getPhysicalAddress()	12
4.5.2.7	loadAndLock() [1/3]	13
4.5.2.8	loadAndLock() [2/3]	13
4.5.2.9	loadAndLock() [3/3]	13
4.5.2.10	resetTable()	14
4.5.2.11	setData()	14
4.5.2.12	unlock()	14
4.6	athena::core::optimizers::AbstractOptimizer Class Reference	15
4.6.1	Detailed Description	15
4.7	athena::core::kernels::AddOpKernel Class Reference	15
4.7.1	Detailed Description	16
4.7.2	Member Function Documentation	16
4.7.2.1	getDerivativeBytecode()	16
4.7.2.2	getDerivativeShape()	17
4.7.2.3	getOperandsCount()	17
4.7.2.4	getOutputShape()	17
4.8	BasicOpCodeParams Struct Reference	18
4.8.1	Detailed Description	18
4.9	athena::backend::generic::CPUDevice Class Reference	18
4.9.1	Detailed Description	18
4.10	athena::core::initializers::DataInitializer Class Reference	19
4.10.1	Detailed Description	19
4.10.2	Member Function Documentation	19
4.10.2.1	initialize()	19
4.11	athena::backend::generic::GenericExecutor Class Reference	20

4.11.1 Detailed Description	20
4.11.2 Member Function Documentation	20
4.11.2.1 execute()	20
4.11.2.2 getMemoryManager()	21
4.12 athena::backend::generic::GenericMemoryManager Class Reference	21
4.12.1 Detailed Description	22
4.12.2 Member Function Documentation	22
4.12.2.1 allocateAndLock() [1/4]	22
4.12.2.2 allocateAndLock() [2/4]	23
4.12.2.3 allocateAndLock() [3/4]	23
4.12.2.4 allocateAndLock() [4/4]	23
4.12.2.5 allocationThreadFunc()	23
4.12.2.6 deinit()	24
4.12.2.7 deleteFromMem()	24
4.12.2.8 getPhysicalAddress()	24
4.12.2.9 init()	25
4.12.2.10 loadAndLock() [1/4]	25
4.12.2.11 loadAndLock() [2/4]	25
4.12.2.12 loadAndLock() [3/4]	26
4.12.2.13 loadAndLock() [4/4]	26
4.12.2.14 processQueueItem()	26
4.12.2.15 setData()	27
4.12.2.16 unlock()	27
4.13 athena::core::optimizers::GradientDescent Class Reference	27
4.13.1 Detailed Description	28
4.13.2 Member Function Documentation	28
4.13.2.1 prepare()	28
4.14 athena::core::InputNode Class Reference	29
4.14.1 Detailed Description	29
4.14.2 Member Function Documentation	29
4.14.2.1 after()	29

4.14.2.2	<code>getData()</code>	30
4.14.2.3	<code>getMappedMemCell()</code>	30
4.14.2.4	<code>isFrozen()</code>	30
4.14.2.5	<code>isInputNode()</code>	30
4.14.2.6	<code>setFrozen()</code>	30
4.14.2.7	<code>setMappedMemCell()</code>	31
4.15	<code>athena::core::kernels::MatMulOpKernel</code> Class Reference	31
4.15.1	Detailed Description	32
4.15.2	Member Function Documentation	32
4.15.2.1	<code>getDerivativeBytecode()</code>	32
4.15.2.2	<code>getDerivativeShape()</code>	32
4.15.2.3	<code>getOperandsCount()</code>	33
4.15.2.4	<code>getOutputShape()</code>	33
4.16	<code>athena::backend::generic::MemoryChunk</code> Struct Reference	34
4.16.1	Detailed Description	34
4.17	<code>athena::core::loss::MSELoss</code> Class Reference	34
4.17.1	Detailed Description	34
4.18	<code>athena::core::loss::MSEOpKernel</code> Class Reference	35
4.18.1	Detailed Description	35
4.18.2	Member Function Documentation	35
4.18.2.1	<code>getDerivativeBytecode()</code>	35
4.18.2.2	<code>getDerivativeShape()</code>	36
4.18.2.3	<code>getOperandsCount()</code>	36
4.18.2.4	<code>getOutputShape()</code>	36
4.19	<code>athena::core::Node</code> Class Reference	37
4.19.1	Detailed Description	38
4.19.2	Member Function Documentation	38
4.19.2.1	<code>after()</code>	38
4.19.2.2	<code>isInputNode()</code>	38
4.20	<code>athena::core::OpKernel</code> Class Reference	39
4.20.1	Detailed Description	39

4.20.2 Member Function Documentation	39
4.20.2.1 getDerivativeBytecode()	39
4.20.2.2 getDerivativeShape()	40
4.20.2.3 getOperandsCount()	40
4.20.2.4 getOutputShape()	41
4.21 athena::backend::generic::QueueItem Struct Reference	42
4.21.1 Detailed Description	42
4.22 athena::core::kernels::ScaleOpKernel Class Reference	42
4.22.1 Detailed Description	43
4.22.2 Member Function Documentation	43
4.22.2.1 getDerivativeBytecode()	43
4.22.2.2 getDerivativeShape()	44
4.22.2.3 getOperandsCount()	44
4.22.2.4 getOutputShape()	44
4.23 athena::core::Session Class Reference	45
4.23.1 Detailed Description	45
4.23.2 Member Function Documentation	45
4.23.2.1 prepare()	45
4.23.2.2 run()	46
4.24 athena::core::kernels::SigmoidOpKernel Class Reference	46
4.24.1 Detailed Description	46
4.24.2 Member Function Documentation	47
4.24.2.1 getDerivativeBytecode()	47
4.24.2.2 getDerivativeShape()	47
4.24.2.3 getOperandsCount()	48
4.24.2.4 getOutputShape()	48
4.25 athena::backend::generic::SwapRecord Struct Reference	48
4.25.1 Detailed Description	49
4.26 athena::core::Tensor Class Reference	49
4.26.1 Detailed Description	49
4.27 athena::core::TensorShape Class Reference	49

4.27.1 Detailed Description	50
4.27.2 Member Function Documentation	50
4.27.2.1 dim()	50
4.27.2.2 dimensions()	50
4.27.2.3 operator"!=()	51
4.27.2.4 operator==()	51
4.27.2.5 totalSize()	51
4.28 athena::backend::VirtualMemory Class Reference	52
4.28.1 Detailed Description	52
4.28.2 Member Function Documentation	52
4.28.2.1 allocate()	52
4.28.2.2 free() [1/2]	53
4.28.2.3 free() [2/2]	53
4.29 athena::backend::VMemoryBlock Struct Reference	53
4.29.1 Detailed Description	53
4.30 athena::backend::VMState Struct Reference	54
4.30.1 Detailed Description	54
4.30.2 Member Data Documentation	54
4.30.2.1 BC	54
4.30.2.2 BP	54
4.30.2.3 IP	54
4.31 athena::core::initializers::VoidInitializer Class Reference	55
4.31.1 Detailed Description	55
4.31.2 Member Function Documentation	55
4.31.2.1 initialize()	55
5 File Documentation	56
5.1 backend/opcodes.h File Reference	56
5.1.1 Detailed Description	56
Index	58

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	18
athena::backend::AbstractExecutor	6
athena::backend::generic::GenericExecutor	20
athena::core::initializers::AbstractInitializer	7
athena::core::initializers::DataInitializer	19
athena::core::initializers::VoidInitializer	55
athena::backend::AbstractMemoryManager	9
athena::backend::generic::GenericMemoryManager	21
athena::core::optimizers::AbstractOptimizer	15
athena::core::optimizers::GradientDescent	27
BasicOpCodeParams	18
athena::backend::generic::MemoryChunk	34
athena::core::Node	37
athena::core::InputNode	29
athena::core::loss::AbstractLossFunction	8
athena::core::loss::MSELoss	34
athena::core::OpKernel	39
athena::core::kernels::AddOpKernel	15
athena::core::kernels::MatMulOpKernel	31
athena::core::kernels::ScaleOpKernel	42
athena::core::kernels::SigmoidOpKernel	46
athena::core::loss::MSEOpKernel	35
athena::backend::generic::QueueItem	42
athena::core::Session	45
athena::backend::generic::SwapRecord	48
athena::core::Tensor	49
athena::core::TensorShape	49
athena::backend::VirtualMemory	52
athena::backend::VMemoryBlock	53
athena::backend::VMState	54

Chapter 2

Class Index

2.1 Class List

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

athena::backend::AbstractDevice	5
athena::backend::AbstractExecutor	6
athena::core::initializers::AbstractInitializer	7
athena::core::loss::AbstractLossFunction	8
athena::backend::AbstractMemoryManager	9
athena::core::optimizers::AbstractOptimizer	15
athena::core::kernels::AddOpKernel	15
BasicOpCodeParams	18
athena::backend::generic::CPUDevice	18
athena::core::initializers::DataInitializer	19
athena::backend::generic::GenericExecutor	20
athena::backend::generic::GenericMemoryManager	21
athena::core::optimizers::GradientDescent	27
athena::core::InputNode	29
athena::core::kernels::MatMulOpKernel	31
athena::backend::generic::MemoryChunk	34
athena::core::loss::MSELoss	34
athena::core::loss::MSEOpKernel	35
athena::core::Node	37
athena::core::OpKernel	39
athena::backend::generic::QueueItem	42
athena::core::kernels::ScaleOpKernel	42
athena::core::Session	45
athena::core::kernels::SigmoidOpKernel	46
athena::backend::generic::SwapRecord	48
athena::core::Tensor	49
athena::core::TensorShape	49
athena::backend::VirtualMemory	52
athena::backend::VMemoryBlock	53
athena::backend::VMState	54
athena::core::initializers::VoidInitializer	55

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

backend/ AbstractDevice.cpp	??
backend/ AbstractDevice.h	??
backend/ AbstractExecutor.cpp	??
backend/ AbstractExecutor.h	??
backend/ AbstractMemoryManager.cpp	??
backend/ AbstractMemoryManager.h	??
backend/ opcode_parser.cpp	??
backend/ opcode_parser.h	??
backend/ opcodes.h	56
backend/ VirtualMemory.cpp	??
backend/ VirtualMemory.h	??
backend/ VMState.h	??
backend/generic/ add.cpp	??
backend/generic/ copy.cpp	??
backend/generic/ CPUDevice.cpp	??
backend/generic/ CPUDevice.h	??
backend/generic/ GenericExecutor.cpp	??
backend/generic/ GenericExecutor.h	??
backend/generic/ GenericMemoryManager.cpp	??
backend/generic/ GenericMemoryManager.h	??
backend/generic/ hadamard.cpp	??
backend/generic/ matmul.cpp	??
backend/generic/ mkscalar.cpp	??
backend/generic/ mse.cpp	??
backend/generic/ mul.cpp	??
backend/generic/ ops.h	??
backend/generic/ scale.cpp	??
backend/generic/ sigmoid.cpp	??
backend/generic/ transpose.cpp	??
core/ DataType.cpp	??
core/ DataType.h	??
core/ InputNode.cpp	??
core/ InputNode.h	??
core/ Node.cpp	??
core/ Node.h	??
core/ OpKernel.cpp	??
core/ OpKernel.h	??
core/ Session.cpp	??

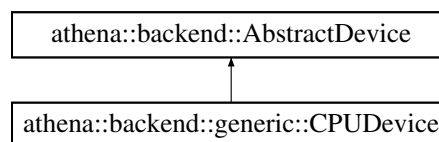
core/ Session.h	??
core/ Tensor.cpp	??
core/ Tensor.h	??
core/ TensorShape.cpp	??
core/ TensorShape.h	??
core/initializers/ AbstractInitializer.cpp	??
core/initializers/ AbstractInitializer.h	??
core/initializers/ DataInitializer.cpp	??
core/initializers/ DataInitializer.h	??
core/initializers/ VoidInitializer.cpp	??
core/initializers/ VoidInitializer.h	??
core/kernels/ AddOpKernel.cpp	??
core/kernels/ AddOpKernel.h	??
core/kernels/ MatMulOpKernel.cpp	??
core/kernels/ MatMulOpKernel.h	??
core/kernels/ ScaleOpKernel.cpp	??
core/kernels/ ScaleOpKernel.h	??
core/kernels/ SigmoidOpKernel.cpp	??
core/kernels/ SigmoidOpKernel.h	??
core/loss/ AbstractLossFunction.cpp	??
core/loss/ AbstractLossFunction.h	??
core/loss/ MSELoss.cpp	??
core/loss/ MSELoss.h	??
core/optimizers/ AbstractOptimizer.cpp	??
core/optimizers/ AbstractOptimizer.h	??
core/optimizers/ GradientDescent.cpp	??
core/optimizers/ GradientDescent.h	??
ops/ add.cpp	??
ops/ add.h	??
ops/ ops.h	??
ops/ sigmoid.cpp	??
ops/ sigmoid.h	??

Chapter 4

Class Documentation

4.1 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**

4.1.1 Detailed Description

Definition at line 20 of file AbstractDevice.h.

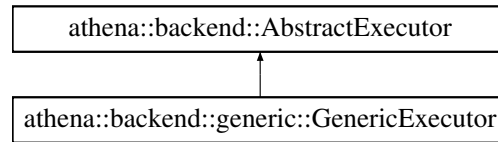
The documentation for this class was generated from the following files:

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

4.2 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**

4.2.1 Detailed Description

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

Definition at line 27 of file `AbstractExecutor.h`.

4.2.2 Member Function Documentation

4.2.2.1 `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`.

4.2.2.2 getMemoryManager()

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

Returns

Memory Manager for current device

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

4.2.2.3 setBytecode()

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

Sets new bytecode

Parameters

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

Definition at line 16 of file AbstractExecutor.cpp.

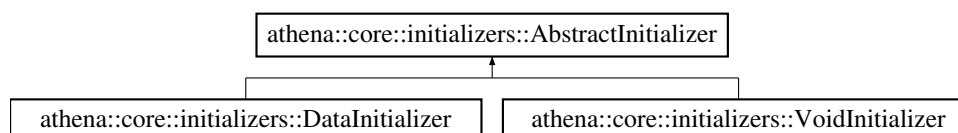
The documentation for this class was generated from the following files:

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

4.3 athena::core::initializers::AbstractInitializer Class Reference

```
#include <AbstractInitializer.h>
```

Inheritance diagram for athena::core::initializers::AbstractInitializer:



Public Member Functions

- virtual void [initialize](#) ([athena::backend::AbstractMemoryManager](#) *manager, [Tensor](#) *tensor)=0

4.3.1 Detailed Description

Initializers help developers load data into corresponding memory type. Every [Tensor](#) is assigned with an Initializer – a subclass of [AbstractInitializer](#). Data will be loaded into memory according to initializer's parameters.

Definition at line 34 of file AbstractInitializer.h.

4.3.2 Member Function Documentation

4.3.2.1 initialize()

```
virtual void athena::core::initializers::AbstractInitializer::initialize (
    athena::backend::AbstractMemoryManager * manager,
    Tensor * tensor ) [pure virtual]
```

Loads data into memory. This method **must not** be called by developers. It is automatically called during initialization process.

Parameters

<i>manager</i>	Memory manager for the current device
<i>tensor</i>	A pointer to a Tensor object, that current initializer is assigned to

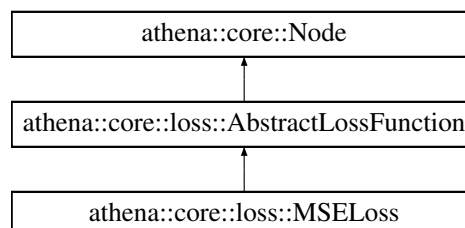
Implemented in [athena::core::initializers::DataInitializer](#), and [athena::core::initializers::VoidInitializer](#).

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

- core/initializers/AbstractInitializer.h

4.4 athena::core::loss::AbstractLossFunction Class Reference

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



Public Member Functions

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

Additional Inherited Members

4.4.1 Detailed Description

Definition at line 21 of file AbstractLossFunction.h.

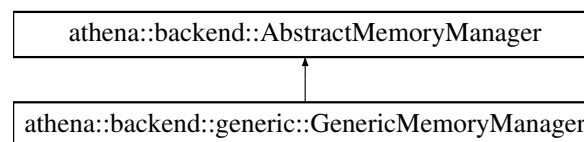
The documentation for this class was generated from the following files:

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

4.5 athena::backend::AbstractMemoryManager Class Reference

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

Inheritance diagram for athena::backend::AbstractMemoryManager:



Public Member Functions

- virtual void **init** ()=0
- virtual void **deinit** ()=0
- void **resetTable** ()
- void **addTensor** (athena::core::Tensor *tensor)
- virtual void * **getPhysicalAddress** (vm_word virtualAddress)=0
- void **loadAndLock** (athena::core::Tensor *tensor)
- void **loadAndLock** (vm_word address)
- virtual void **loadAndLock** (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)
- void **allocateAndLock** (athena::core::Tensor *tensor)
- void **allocateAndLock** (vm_word address)
- virtual void **allocateAndLock** (vm_word address, unsigned long length)=0
- virtual void **setData** (vm_word tensorAddress, vm_word offset, vm_word length, void *data)=0
- virtual void **getData** (vm_word tensorAddress, vm_word offset, vm_word length, void *data)=0

Protected Attributes

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

4.5.1 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.

Definition at line 30 of file AbstractMemoryManager.h.

4.5.2 Member Function Documentation

4.5.2.1 addTensor()

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

Adds Tensor to table

Parameters

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

Definition at line 20 of file AbstractMemoryManager.cpp.

4.5.2.2 allocateAndLock() [1/3]

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

Allocate space in the fastest memory available without loading any data and lock it (prevent from being of-floated)

Parameters

<i>tensor</i>	Tensor memory is being allocated for
---------------	--------------------------------------

Definition at line 54 of file AbstractMemoryManager.cpp.

Referenced by [athena::core::initializers::DataInitializer::initialize\(\)](#).

4.5.2.3 allocateAndLock() [2/3]

```
void athena::backend::AbstractMemoryManager::allocateAndLock (
    vm_word address )
```

Allocate space in the fastest memory available without loading any data and lock it (prevent from being of-loaded)

Parameters

<i>address</i>	Virtual address of Tensor memory is being allocated for
----------------	---

Definition at line 60 of file AbstractMemoryManager.cpp.

4.5.2.4 allocateAndLock() [3/3]

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

Allocate space in the fastest memory available without loading any data and lock it (prevent from being of-floaded)

Parameters

<i>address</i>	Virtual address of Tensor memory is being allocated for
<i>length</i>	Length in bytes for the piece of memory that is being allocated

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

4.5.2.5 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](#).

4.5.2.6 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 $2^{64}-1$
-----------------------	---

Returns

Pointer to physical memory

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

4.5.2.7 loadAndLock() [1/3]

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

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

Parameters

<i>tensor</i>	Tensor containing data
---------------	------------------------

Definition at line 25 of file AbstractMemoryManager.cpp.

4.5.2.8 loadAndLock() [2/3]

```
void athena::backend::AbstractMemoryManager::loadAndLock (
    vm_word address )
```

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 of Tensor containing data
----------------	---

Definition at line 30 of file AbstractMemoryManager.cpp.

4.5.2.9 loadAndLock() [3/3]

```
virtual void athena::backend::AbstractMemoryManager::loadAndLock (
    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](#).

4.5.2.10 resetTable()

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

Clears table of Tensors

Definition at line 16 of file AbstractMemoryManager.cpp.

4.5.2.11 setData()

```
virtual void athena::backend::AbstractMemoryManager::setData (
    vm_word tensorAddress,
    vm_word offset,
    vm_word length,
    void * data ) [pure virtual]
```

Sets memory with the data

Parameters

<i>tensorAddress</i>	Virtual address of Tensor beginning (see Tensor::getStartAddress)
<i>offset</i>	Offset in bytes from the beginning
<i>length</i>	Length of piece of data in bytes
<i>data</i>	Pointer to data

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

Referenced by [athena::core::initializers::DataInitializer::initialize\(\)](#).

4.5.2.12 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](#).

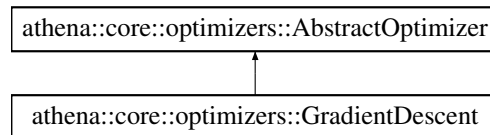
Referenced by [athena::core::initializers::DataInitializer::initialize\(\)](#).

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

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

4.6 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**

4.6.1 Detailed Description

Definition at line 24 of file AbstractOptimizer.h.

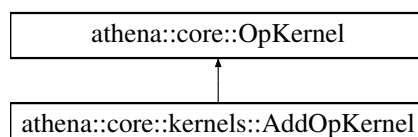
The documentation for this class was generated from the following files:

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

4.7 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

4.7.1 Detailed Description

Performs sum of 2 given Tensors

Definition at line 24 of file AddOpKernel.h.

4.7.2 Member Function Documentation

4.7.2.1 getDerivativeBytecode()

```
std::vector< vm_word > 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`.

Definition at line 39 of file AddOpKernel.cpp.

4.7.2.2 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](#).

Definition at line 61 of file AddOpKernel.cpp.

4.7.2.3 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](#).

Definition at line 18 of file AddOpKernel.cpp.

4.7.2.4 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](#).

Definition at line 56 of file AddOpKernel.cpp.

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

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

4.8 BasicOpCodeParams Struct Reference

4.8.1 Detailed Description

Definition at line 141 of file opcodes.h.

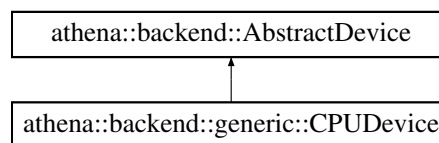
The documentation for this struct was generated from the following file:

- backend/opcodes.h

4.9 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

4.9.1 Detailed Description

This class represents a CPU It encapsulates Memory Manager

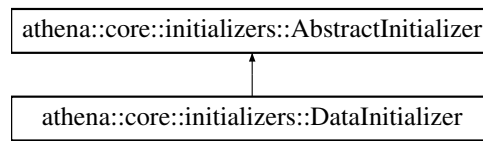
Definition at line 27 of file CPUDevice.h.

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

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

4.10 athena::core::initializers::DataInitializer Class Reference

Inheritance diagram for athena::core::initializers::DataInitializer:



Public Member Functions

- **DataInitializer** (const [DataInitializer](#) &src)
- [DataInitializer](#) & **operator=** (const [DataInitializer](#) &src)
- void **setData** (void *ptr, size_t length)
- void **initialize** ([athena::backend::AbstractMemoryManager](#) *manager, [Tensor](#) *tensor) override

4.10.1 Detailed Description

Definition at line 23 of file DataInitializer.h.

4.10.2 Member Function Documentation

4.10.2.1 initialize()

```

void athena::core::initializers::DataInitializer::initialize (
    athena::backend::AbstractMemoryManager * manager,
    Tensor * tensor ) [override], [virtual]
  
```

Loads data into memory. This method **must not** be called by developers. It is automatically called during initialization process.

Parameters

<i>manager</i>	Memory manager for the current device
<i>tensor</i>	A pointer to a Tensor object, that current initializer is assigned to

Implements [athena::core::initializers::AbstractInitializer](#).

Definition at line 16 of file DataInitializer.cpp.

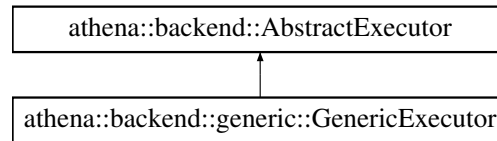
The documentation for this class was generated from the following files:

- core/initializers/DataInitializer.h
- core/initializers/DataInitializer.cpp

4.11 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

4.11.1 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.

Definition at line 50 of file [GenericExecutor.h](#).

4.11.2 Member Function Documentation

4.11.2.1 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](#).

Definition at line 22 of file [GenericExecutor.cpp](#).

4.11.2.2 getMemoryManager()

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

Returns

Memory Manager for current device

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

Definition at line 31 of file GenericExecutor.cpp.

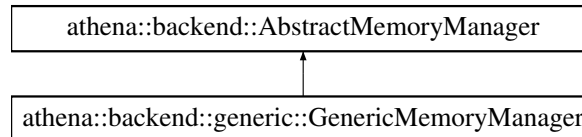
The documentation for this class was generated from the following files:

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

4.12 athena::backend::generic::GenericMemoryManager Class Reference

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

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



Public Member Functions

- void **init** () override
- void **deinit** () override
- void * **getPhysicalAddress** (vm_word virtualAddress) override
- void **loadAndLock** (vm_word address, unsigned long length) override
- void **allocateAndLock** (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 **setData** (vm_word tensorAddress, vm_word offset, vm_word length, void *data) override
- void **getData** (vm_word tensorAddress, vm_word offset, vm_word length, void *data) override
- void **loadAndLock** (athena::core::Tensor *tensor)
- void **loadAndLock** (vm_word address)
- virtual void **loadAndLock** (vm_word address, unsigned long length)=0
- void **allocateAndLock** (athena::core::Tensor *tensor)
- void **allocateAndLock** (vm_word address)
- virtual void **allocateAndLock** (vm_word address, unsigned long length)=0

Protected Member Functions

- void [allocationThreadFunc](#) (int laneId)
- void [processQueueItem](#) ([QueueItem](#) *item)

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**

4.12.1 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.

Definition at line 76 of file [GenericMemoryManager.h](#).

4.12.2 Member Function Documentation

4.12.2.1 [allocateAndLock\(\)](#) [1/4]

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

Allocate space in the fastest memory available without loading any data and lock it (prevent from being of-loaded)

Parameters

<i>address</i>	Virtual address of Tensor memory is being allocated for
<i>length</i>	Length in bytes for the piece of memory that is being allocated

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

Definition at line 174 of file [GenericMemoryManager.cpp](#).

4.12.2.2 allocateAndLock() [2/4]

```
void athena::backend::AbstractMemoryManager::allocateAndLock
```

Allocate space in the fastest memory available without loading any data and lock it (prevent from being of-loaded)

Parameters

<i>tensor</i>	Tensor memory is being allocated for
---------------	--------------------------------------

Definition at line 54 of file AbstractMemoryManager.cpp.

4.12.2.3 allocateAndLock() [3/4]

```
void athena::backend::AbstractMemoryManager::allocateAndLock
```

Allocate space in the fastest memory available without loading any data and lock it (prevent from being of-loaded)

Parameters

<i>address</i>	Virtual address of Tensor memory is being allocated for
----------------	---

Definition at line 60 of file AbstractMemoryManager.cpp.

4.12.2.4 allocateAndLock() [4/4]

```
virtual void athena::backend::AbstractMemoryManager::allocateAndLock
```

Allocate space in the fastest memory available without loading any data and lock it (prevent from being of-loaded)

Parameters

<i>address</i>	Virtual address of Tensor memory is being allocated for
<i>length</i>	Length in bytes for the piece of memory that is being allocated

4.12.2.5 allocationThreadFunc()

```
void athena::backend::generic::GenericMemoryManager::allocationThreadFunc (
    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>	

Definition at line 37 of file GenericMemoryManager.cpp.

Referenced by `init()`.

4.12.2.6 `deinit()`

```
void athena::backend::generic::GenericMemoryManager::deinit ( ) [override], [virtual]
```

Free RAM and stop all threads-memory lanes

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

Definition at line 84 of file GenericMemoryManager.cpp.

4.12.2.7 `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](#).

Definition at line 148 of file GenericMemoryManager.cpp.

4.12.2.8 `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](#).

Definition at line 51 of file GenericMemoryManager.cpp.

4.12.2.9 init()

```
void athena::backend::generic::GenericMemoryManager::init ( ) [override], [virtual]
```

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

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

Definition at line 18 of file GenericMemoryManager.cpp.

4.12.2.10 loadAndLock() [1/4]

```
void athena::backend::generic::GenericMemoryManager::loadAndLock (
    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](#).

Definition at line 65 of file GenericMemoryManager.cpp.

4.12.2.11 loadAndLock() [2/4]

```
void athena::backend::AbstractMemoryManager::loadAndLock
```

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 of Tensor containing data
----------------	---

Definition at line 30 of file AbstractMemoryManager.cpp.

4.12.2.12 loadAndLock() [3/4]

```
void athena::backend::AbstractMemoryManager::loadAndLock
```

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

Parameters

<i>tensor</i>	Tensor containing data
---------------	------------------------

Definition at line 25 of file AbstractMemoryManager.cpp.

4.12.2.13 loadAndLock() [4/4]

```
virtual void athena::backend::AbstractMemoryManager::loadAndLock
```

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

4.12.2.14 processQueueItem()

```
void athena::backend::generic::GenericMemoryManager::processQueueItem (
    QueueItem * item ) [protected]
```

This method does actual physical memory allocation

Parameters

<i>item</i>	
-------------	--

Definition at line 219 of file GenericMemoryManager.cpp.

4.12.2.15 setData()

```
void athena::backend::generic::GenericMemoryManager::setData (
    vm_word tensorAddress,
    vm_word offset,
    vm_word length,
    void * data ) [override], [virtual]
```

Sets memory with the data

Parameters

<i>tensorAddress</i>	Virtual address of Tensor beginning (see Tensor::getStartAddress)
<i>offset</i>	Offset in bytes from the beginning
<i>length</i>	Length of piece of data in bytes
<i>data</i>	Pointer to data

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

Definition at line 195 of file GenericMemoryManager.cpp.

4.12.2.16 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](#).

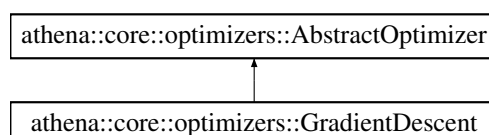
Definition at line 105 of file GenericMemoryManager.cpp.

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

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

4.13 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< vm_word >, vm_word > **getByteCode** ([athena::core::loss::AbstractLossFunction](#) *node)

Protected Attributes

- float **learningRate**

4.13.1 Detailed Description

Definition at line 21 of file GradientDescent.h.

4.13.2 Member Function Documentation

4.13.2.1 prepare()

```
void athena::core::optimizers::GradientDescent::prepare ( ) [override], [virtual]
```

Generate bytecode for backpropagation

The whole algorithm:

1. Calculate actual error E
2. Let's Q - queue of nodes, EQ - queue of errors
 - (a) $node \rightarrow Q$
 - (b) $E \rightarrow EQ$
3. For each node N in Q
 - (a) $EQ \rightarrow E$
 - (b) If this is variable node, adjust weights: $N = N - \alpha * E$
 - (c) If this is regular node, for each incoming node I:
 - i. If I is constant, skip
 - ii. $E_i = D_i \odot E$, where E_i is the new error value, D_i - derivative of N with respect to I, \odot - Hadamard (elementwise) product.
 - iii. $E_i \rightarrow EQ$
 - iv. $I \rightarrow Q$

Implements [athena::core::optimizers::AbstractOptimizer](#).

Definition at line 24 of file GradientDescent.cpp.

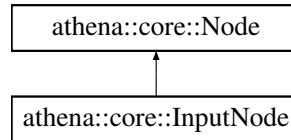
The documentation for this class was generated from the following files:

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

4.14 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)
- void **setInitializer** ([athena::core::initializers::AbstractInitializer](#) *initializer)
- [athena::core::initializers::AbstractInitializer](#) * **getInitializer** ()
- bool [isGarbage](#) () override

Additional Inherited Members

4.14.1 Detailed Description

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

Definition at line 26 of file InputNode.h.

4.14.2 Member Function Documentation

4.14.2.1 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](#).

Definition at line 63 of file InputNode.h.

4.14.2.2 getData()

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

Get data associated with this [InputNode](#)

Returns

Pointer to [Tensor](#)

Definition at line 28 of file InputNode.cpp.

4.14.2.3 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

Definition at line 24 of file InputNode.cpp.

4.14.2.4 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

Definition at line 32 of file InputNode.cpp.

4.14.2.5 isInputNode()

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

Check if it is an input node

Returns

true

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

Definition at line 16 of file InputNode.cpp.

4.14.2.6 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)
---------------	--

Definition at line 36 of file InputNode.cpp.

4.14.2.7 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
-------------	--------------------

Definition at line 20 of file InputNode.cpp.

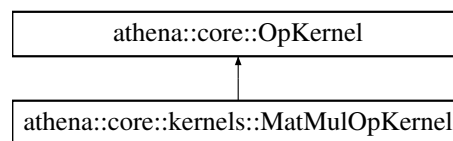
The documentation for this class was generated from the following files:

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

4.15 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

4.15.1 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)

Definition at line 29 of file MatMulOpKernel.h.

4.15.2 Member Function Documentation

4.15.2.1 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](#).

Definition at line 38 of file MatMulOpKernel.cpp.

4.15.2.2 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](#).

Definition at line 63 of file MatMulOpKernel.cpp.

4.15.2.3 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](#).

Definition at line 16 of file MatMulOpKernel.cpp.

4.15.2.4 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](#).

Definition at line 56 of file MatMulOpKernel.cpp.

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

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

4.16 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**

4.16.1 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.

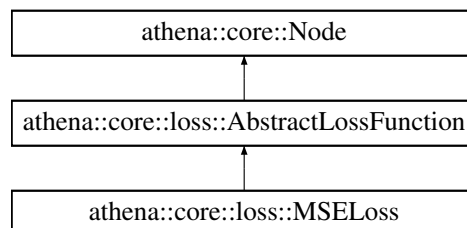
Definition at line 46 of file GenericMemoryManager.h.

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

- backend/generic/GenericMemoryManager.h

4.17 athena::core::loss::MSELoss Class Reference

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



Additional Inherited Members

4.17.1 Detailed Description

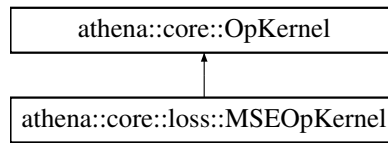
Definition at line 47 of file MSELoss.h.

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

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

4.18 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< 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

4.18.1 Detailed Description

Definition at line 22 of file MSELoss.h.

4.18.2 Member Function Documentation

4.18.2.1 getDerivativeBytecode()

```

std::vector< vm_word > athena::core::loss::MSEOpKernel::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](#).

Definition at line 40 of file MSELoss.cpp.

4.18.2.2 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](#).

Definition at line 59 of file MSELoss.cpp.

4.18.2.3 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](#).

Definition at line 22 of file MSELoss.cpp.

4.18.2.4 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](#).

Definition at line 53 of file MSELoss.cpp.

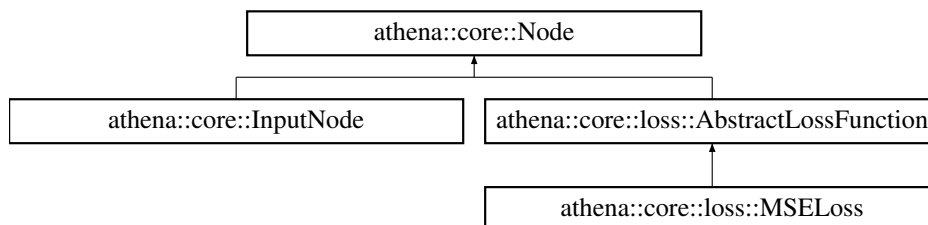
The documentation for this class was generated from the following files:

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

4.19 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)
- vm_word [getDerivative](#) (unsigned long i)
- void [setCalculated](#) (unsigned long resCell)
- bool [isCalculated](#) ()
- vm_word [getResult](#) ()
- void [updateUsageCount](#) ()
- virtual bool [isGarbage](#) ()
- void [setPersistResult](#) ()

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` **persistResult**

4.19.1 Detailed Description

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

Definition at line 29 of file Node.h.

4.19.2 Member Function Documentation

4.19.2.1 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](#).

Definition at line 22 of file Node.cpp.

4.19.2.2 isInputNode()

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

Check if it is an input node

Returns

false

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

Definition at line 33 of file Node.cpp.

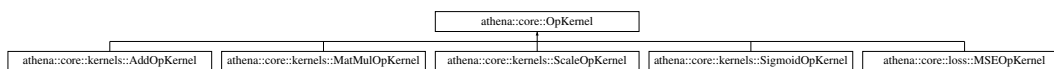
The documentation for this class was generated from the following files:

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

4.20 athena::core::OpKernel Class Reference

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

Inheritance diagram for athena::core::OpKernel:



Public Member Functions

- **OpKernel** (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

- std::string **name**

4.20.1 Detailed Description

Operation skeleton Each operation has OpCode

Definition at line 29 of file OpKernel.h.

4.20.2 Member Function Documentation

4.20.2.1 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

Returns

Implemented in `athena::core::kernels::MatMulOpKernel`, `athena::core::kernels::SigmoidOpKernel`, `athena::core::loss::MSEOpKernel`, `athena::core::kernels::AddOpKernel`, and `athena::core::kernels::ScaleOpKernel`.

4.20.2.2 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::kernels::AddOpKernel`, `athena::core::kernels::ScaleOpKernel`, and `athena::core::loss::MSEOpKernel`.

4.20.2.3 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::kernels::AddOpKernel`, `athena::core::kernels::ScaleOpKernel`, and `athena::core::loss::MSEOpKernel`.

4.20.2.4 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::kernels::AddOpKernel](#), [athena::core::kernels::ScaleOpKernel](#), and [athena::core::loss::MSEOpKernel](#).

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

- [core/OpKernel.h](#)

4.21 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

4.21.1 Detailed Description

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

Definition at line 60 of file [GenericMemoryManager.h](#).

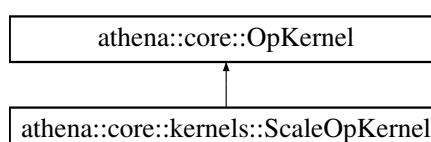
The documentation for this struct was generated from the following file:

- [backend/generic/GenericMemoryManager.h](#)

4.22 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

4.22.1 Detailed Description

Multiply [Tensor](#) by scalar

Definition at line 24 of file `ScaleOpKernel.h`.

4.22.2 Member Function Documentation

4.22.2.1 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](#).

Definition at line 35 of file `ScaleOpKernel.cpp`.

4.22.2.2 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](#).

Definition at line 53 of file ScaleOpKernel.cpp.

4.22.2.3 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](#).

Definition at line 16 of file ScaleOpKernel.cpp.

4.22.2.4 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](#).

Definition at line 47 of file ScaleOpKernel.cpp.

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

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

4.23 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)
- [athena::backend::AbstractExecutor](#) * [getExecutor](#) ()
- [athena::backend::VirtualMemory](#) * [getMemory](#) ()

4.23.1 Detailed Description

The class encapsulates everything needed for a single training step

Definition at line 28 of file Session.h.

4.23.2 Member Function Documentation

4.23.2.1 prepare()

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

Generates bytecode for the whole graph

Parameters

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

Definition at line 18 of file Session.cpp.

4.23.2.2 run()

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

does single training step

Returns

result tensor

Definition at line 126 of file Session.cpp.

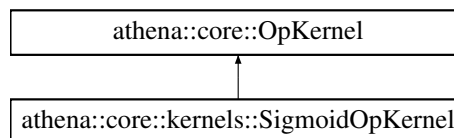
The documentation for this class was generated from the following files:

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

4.24 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

4.24.1 Detailed Description

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

Definition at line 26 of file SigmoidOpKernel.h.

4.24.2 Member Function Documentation

4.24.2.1 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](#).

Definition at line 38 of file SigmoidOpKernel.cpp.

4.24.2.2 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](#).

Definition at line 53 of file SigmoidOpKernel.cpp.

4.24.2.3 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](#).

Definition at line 19 of file SigmoidOpKernel.cpp.

4.24.2.4 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](#).

Definition at line 59 of file SigmoidOpKernel.cpp.

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

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

4.25 athena::backend::generic::SwapRecord Struct Reference

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

Public Attributes

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

4.25.1 Detailed Description

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

Definition at line 34 of file GenericMemoryManager.h.

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

- backend/generic/GenericMemoryManager.h

4.26 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)

4.26.1 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.

Definition at line 39 of file Tensor.h.

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

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

4.27 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

4.27.1 Detailed Description

Class represents size parameters for [Tensor](#)

Definition at line 25 of file [TensorShape.h](#).

4.27.2 Member Function Documentation

4.27.2.1 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

Definition at line 33 of file [TensorShape.cpp](#).

Referenced by [operator==\(\)](#).

4.27.2.2 dimensions()

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

Returns

Number of dimensions in [Tensor](#)

Definition at line 29 of file TensorShape.cpp.

Referenced by operator==().

4.27.2.3 operator!=()

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

Parameters

<i>rhs</i>	TensorShape to be compared with
------------	---

Returns

True if dimensions are different, else False

Definition at line 58 of file TensorShape.cpp.

4.27.2.4 operator==()

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

Returns

True if dimensions are equal, else False

Definition at line 42 of file TensorShape.cpp.

4.27.2.5 totalSize()

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

Returns

Total number of elements in [Tensor](#)

Definition at line 17 of file TensorShape.cpp.

Referenced by `athena::backend::VirtualMemory::allocate()`, `athena::backend::AbstractMemoryManager::allocateAndLock()`, and `athena::backend::AbstractMemoryManager::loadAndLock()`.

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

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

4.28 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)`
- `athena::core::Tensor * getTensor (vm_word address)`

4.28.1 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](#)

Definition at line 45 of file VirtualMemory.h.

4.28.2 Member Function Documentation

4.28.2.1 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

Definition at line 28 of file VirtualMemory.cpp.

4.28.2.2 free() [1/2]

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

Marks memory as free

Parameters

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

Definition at line 137 of file VirtualMemory.cpp.

4.28.2.3 free() [2/2]

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

Marks memory as free

Parameters

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

Definition at line 97 of file VirtualMemory.cpp.

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

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

4.29 athena::backend::VMemoryBlock Struct Reference

Public Attributes

- bool **isUsed**
- vm_word **startAddress**
- vm_word **endAddress**
- VMemoryBlock * **nextBlock**
- VMemoryBlock * **prevBlock**

4.29.1 Detailed Description

Definition at line 23 of file VirtualMemory.h.

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

- backend/VirtualMemory.h

4.30 athena::backend::VMState Struct Reference

```
#include <VMState.h>
```

Public Attributes

- unsigned long [BC](#) = 1
- unsigned long [BP](#) = 0
- unsigned long [IP](#) = 0

4.30.1 Detailed Description

[VMState](#) structure describes current state of executor thread

Definition at line 22 of file [VMState.h](#).

4.30.2 Member Data Documentation

4.30.2.1 BC

```
unsigned long athena::backend::VMState::BC = 1
```

Batch Counter, number of batches

Definition at line 27 of file [VMState.h](#).

4.30.2.2 BP

```
unsigned long athena::backend::VMState::BP = 0
```

Batch Pointer, number of current batch

Definition at line 32 of file [VMState.h](#).

4.30.2.3 IP

```
unsigned long athena::backend::VMState::IP = 0
```

Instruction Pointer, points to instruction that is going to be executed

Definition at line 37 of file [VMState.h](#).

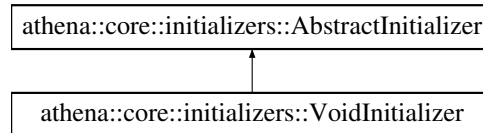
The documentation for this struct was generated from the following file:

- [backend/VMState.h](#)

4.31 athena::core::initializers::VoidInitializer Class Reference

```
#include <VoidInitializer.h>
```

Inheritance diagram for athena::core::initializers::VoidInitializer:



Public Member Functions

- void `initialize` (athena::backend::AbstractMemoryManager *, Tensor *) override

4.31.1 Detailed Description

`VoidInitializer` is the simplest initializer possible. It does not initialize `Tensor` in any way. It is the default initializer for newly created `Tensor`

Definition at line 25 of file `VoidInitializer.h`.

4.31.2 Member Function Documentation

4.31.2.1 initialize()

```
void athena::core::initializers::VoidInitializer::initialize (
    athena::backend::AbstractMemoryManager * manager,
    Tensor * tensor ) [override], [virtual]
```

Loads data into memory. This method **must not** be called by developers. It is automatically called during initialization process.

Parameters

<i>manager</i>	Memory manager for the current device
<i>tensor</i>	A pointer to a <code>Tensor</code> object, that current initializer is assigned to

Implements `athena::core::initializers::AbstractInitializer`.

Definition at line 16 of file `VoidInitializer.cpp`.

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

- `core/initializers/VoidInitializer.h`
- `core/initializers/VoidInitializer.cpp`

Chapter 5

File Documentation

5.1 backend/opcodes.h File Reference

Classes

- struct `BasicOpCodeParams`

Typedefs

- typedef unsigned long `vm_word`

Enumerations

- enum `OpCode` : `vm_word` {
 DEL = 0x000000000,
 PUSH = 0x010000000,
 POP = 0x020000000,
 JMP = 0x030000000,
 ADD = 0x040000000,
 MATMUL = 0x050000000,
 MKSCALAR = 0x060000000,
 SCALE = 0x070000000,
 SIGMOID = 0x080000000,
 SIGMOID_DERIV = 0x090000000,
 TRANPOSE = 0x0A0000000,
 COPY = 0x0B0000000,
 MSE = 0x0C0000000,
 MSE_DERIV = 0x0D0000000,
 ALLOC = 0x0E0000000,
 MUL = 0x0F0000000,
 HADAMARD = 0x100000000 }

5.1.1 Detailed Description

`OpCode` specifies digital codes for Athena VM's operations. Operation is encoded with operation code and its parameters. Bits 0..31 (assuming little-endian) are used by operation code. Bits 32..63 are used by operation parameters.

Description of Athena VM operations

Operation	Description	Parameters	Arguments
ADD	Produces sum of two Tensors		3 arguments: pointers to two summands and result Tensor
MATMUL	Produces product of two matrix Tensors	todo describe 2 parameters: transpose args	3 arguments: pointers to two matrices and result Tensor
MKSCALAR	Creates Tensor from scalar constant	todo describe 1 parameter: data type	2 arguments: scalar and result Tensor
SCALE	Multiplies every Tensor element by scalar	No parameters	3 arguments: pointers to scalar Tensor, source Tensor and result Tensor
SIGMOID	Applies sigmoid function to every element of Tensor	No parameters	2 arguments: source Tensor and result Tensor
SIGMOID_DERIV	Calculates derivative of sigmoid function	No parameters	2 arguments: source Tensor and result Tensor
TRANSPOSE	Transpose matrix Tensor	No parameters	2 arguments: source Tensor and result Tensor
TRANSPOSE	Transpose matrix Tensor	No parameters	2 arguments: source Tensor and result Tensor
COPY	Copy Tensor	No parameters	2 arguments: source Tensor and result Tensor
MSE	Calculate minimal squared error	No parameters	3 arguments: produced Tensor, label Tensor and result Tensor
MSE_DERIV	Calculate derivative of minimal squared error	No parameters	3 arguments: produced Tensor, label Tensor and result Tensor
ALLOC	Allocate memory for Tensor	todo describe one parameter: data type	First argument always specifies number of dimensions, last arg specifies virtual address of new Tensor.
MUL	Multiply two Tensors	No parameters	3 arguments: multiplier Tensors and result Tensor
HADAMARD	Do element-wise multiplication of two matrices	No parameters	3 arguments: multiplier Tensors and result Tensor

Index

- addTensor
 - athena::backend::AbstractMemoryManager, 10
- after
 - athena::core::InputNode, 29
 - athena::core::Node, 38
- allocate
 - athena::backend::VirtualMemory, 52
- allocateAndLock
 - athena::backend::AbstractMemoryManager, 10, 12
 - athena::backend::generic::GenericMemoryManager, 22, 23
- allocationThreadFunc
 - athena::backend::generic::GenericMemoryManager, 23
- athena::backend::AbstractDevice, 5
- athena::backend::AbstractExecutor, 6
 - execute, 6
 - getMemoryManager, 6
 - setBytecode, 7
- athena::backend::AbstractMemoryManager, 9
 - addTensor, 10
 - allocateAndLock, 10, 12
 - deleteFromMem, 12
 - getPhysicalAddress, 12
 - loadAndLock, 13
 - resetTable, 14
 - setData, 14
 - unlock, 14
- athena::backend::VMState, 54
 - BC, 54
 - BP, 54
 - IP, 54
- athena::backend::VMemoryBlock, 53
- athena::backend::VirtualMemory, 52
 - allocate, 52
 - free, 52, 53
- athena::backend::generic::CPUDevice, 18
- athena::backend::generic::GenericExecutor, 20
 - execute, 20
 - getMemoryManager, 20
- athena::backend::generic::GenericMemoryManager, 21
 - allocateAndLock, 22, 23
 - allocationThreadFunc, 23
 - deinit, 24
 - deleteFromMem, 24
 - getPhysicalAddress, 24
 - init, 25
 - loadAndLock, 25, 26
 - processQueueItem, 26
 - setData, 26
 - unlock, 27
- athena::backend::generic::MemoryChunk, 34
- athena::backend::generic::QueueItem, 42
- athena::backend::generic::SwapRecord, 48
- athena::core::InputNode, 29
 - after, 29
 - getData, 29
 - getMappedMemCell, 30
 - isFrozen, 30
 - isInputNode, 30
 - setFrozen, 30
 - setMappedMemCell, 31
- athena::core::Node, 37
 - after, 38
 - isInputNode, 38
- athena::core::OpKernel, 39
 - getDerivativeBytecode, 39
 - getDerivativeShape, 40
 - getOperandsCount, 40
 - getOutputShape, 40
- athena::core::Session, 45
 - prepare, 45
 - run, 46
- athena::core::Tensor, 49
- athena::core::TensorShape, 49
 - dim, 50
 - dimensions, 50
 - operator!=, 51
 - operator==, 51
 - totalSize, 51
- athena::core::initializers::AbstractInitializer, 7
 - initialize, 8
- athena::core::initializers::DataInitializer, 19
 - initialize, 19
- athena::core::initializers::VoidInitializer, 55
 - initialize, 55
- athena::core::kernels::AddOpKernel, 15
 - getDerivativeBytecode, 16
 - getDerivativeShape, 16
 - getOperandsCount, 17
 - getOutputShape, 17
- athena::core::kernels::MatMulOpKernel, 31
 - getDerivativeBytecode, 32
 - getDerivativeShape, 32
 - getOperandsCount, 33
 - getOutputShape, 33
- athena::core::kernels::ScaleOpKernel, 42
 - getDerivativeBytecode, 43
 - getDerivativeShape, 43
 - getOperandsCount, 44
 - getOutputShape, 44
- athena::core::kernels::SigmoidOpKernel, 46

- getDerivativeBytecode, 47
- getDerivativeShape, 47
- getOperandsCount, 47
- getOutputShape, 48
- athena::core::loss::AbstractLossFunction, 8
- athena::core::loss::MSELoss, 34
- athena::core::loss::MSEOpKernel, 35
 - getDerivativeBytecode, 35
 - getDerivativeShape, 36
 - getOperandsCount, 36
 - getOutputShape, 36
- athena::core::optimizers::AbstractOptimizer, 15
- athena::core::optimizers::GradientDescent, 27
 - prepare, 28
- backend/opcodes.h, 56
- BasicOpCodeParams, 18
- BC
 - athena::backend::VMState, 54
- BP
 - athena::backend::VMState, 54
- deinit
 - athena::backend::generic::GenericMemory↔
Manager, 24
- deleteFromMem
 - athena::backend::AbstractMemoryManager, 12
 - athena::backend::generic::GenericMemory↔
Manager, 24
- dim
 - athena::core::TensorShape, 50
- dimensions
 - athena::core::TensorShape, 50
- execute
 - athena::backend::AbstractExecutor, 6
 - athena::backend::generic::GenericExecutor, 20
- free
 - athena::backend::VirtualMemory, 52, 53
- getData
 - athena::core::InputNode, 29
- getDerivativeBytecode
 - athena::core::OpKernel, 39
 - athena::core::kernels::AddOpKernel, 16
 - athena::core::kernels::MatMulOpKernel, 32
 - athena::core::kernels::ScaleOpKernel, 43
 - athena::core::kernels::SigmoidOpKernel, 47
 - athena::core::loss::MSEOpKernel, 35
- getDerivativeShape
 - athena::core::OpKernel, 40
 - athena::core::kernels::AddOpKernel, 16
 - athena::core::kernels::MatMulOpKernel, 32
 - athena::core::kernels::ScaleOpKernel, 43
 - athena::core::kernels::SigmoidOpKernel, 47
 - athena::core::loss::MSEOpKernel, 36
- getMappedMemCell
 - athena::core::InputNode, 30
- getMemoryManager
 - athena::backend::AbstractExecutor, 6
 - athena::backend::generic::GenericExecutor, 20
- getOperandsCount
 - athena::core::OpKernel, 40
 - athena::core::kernels::AddOpKernel, 17
 - athena::core::kernels::MatMulOpKernel, 33
 - athena::core::kernels::ScaleOpKernel, 44
 - athena::core::kernels::SigmoidOpKernel, 47
 - athena::core::loss::MSEOpKernel, 36
- getOutputShape
 - athena::core::OpKernel, 40
 - athena::core::kernels::AddOpKernel, 17
 - athena::core::kernels::MatMulOpKernel, 33
 - athena::core::kernels::ScaleOpKernel, 44
 - athena::core::kernels::SigmoidOpKernel, 48
 - athena::core::loss::MSEOpKernel, 36
- getPhysicalAddress
 - athena::backend::AbstractMemoryManager, 12
 - athena::backend::generic::GenericMemory↔
Manager, 24
- init
 - athena::backend::generic::GenericMemory↔
Manager, 25
- initialize
 - athena::core::initializers::AbstractInitializer, 8
 - athena::core::initializers::DataInitializer, 19
 - athena::core::initializers::VoidInitializer, 55
- IP
 - athena::backend::VMState, 54
- isFrozen
 - athena::core::InputNode, 30
- isInputNode
 - athena::core::InputNode, 30
 - athena::core::Node, 38
- loadAndLock
 - athena::backend::AbstractMemoryManager, 13
 - athena::backend::generic::GenericMemory↔
Manager, 25, 26
- operator!=
 - athena::core::TensorShape, 51
- operator==
 - athena::core::TensorShape, 51
- prepare
 - athena::core::Session, 45
 - athena::core::optimizers::GradientDescent, 28
- processQueueItem
 - athena::backend::generic::GenericMemory↔
Manager, 26
- resetTable
 - athena::backend::AbstractMemoryManager, 14
- run
 - athena::core::Session, 46
- setBytecode
 - athena::backend::AbstractExecutor, 7
- setData
 - athena::backend::AbstractMemoryManager, 14
 - athena::backend::generic::GenericMemory↔
Manager, 26

setFrozen
 athena::core::InputNode, [30](#)
setMappedMemCell
 athena::core::InputNode, [31](#)

totalSize
 athena::core::TensorShape, [51](#)

unlock
 athena::backend::AbstractMemoryManager, [14](#)
 athena::backend::generic::GenericMemory↵
 Manager, [27](#)