ISBN 978-89-6211-698-4

대용량 데이터 가시화를 위한 애니메이션 캐쉬 알고리즘 설계 및 구현

(Animation cache algorithm design and implementation for massive data visualization)

김 민 아 (petimina@kisti.re.kr)

한 국 과 학 기 술 정 보 연 구 원
Korea Institute of Science & Technology Information

목차

그림 차례

표 차례

VISE는 병렬 렌더링을 지원하지 않는다. 동일한 구조인 paraview 에서 대용량 데이터 가시화를 위한 애니메이션을 수행한 결과는 GLOVE의 성능이 paraview 의 paraview 보다 125배 빠른 것을 보여 준다.

[표 1] 1.2TB, 90 time step 로터 시뮬레이션 데이터 성능 측정

Tool	Parallel	COVISE	Paraview	GLOVE	COVISE:Paraview:GLOVE
Probe by Plane (sec)	single	36	11.93	20.04	1.8 : 0.59 : 1
	multi	N/A	1.79	1.80	∞ : 0.99 : 1
iso-surface (sec)	single	47	14.24	18.21	2.58: 0.78 : 1
	multi	N/A	10.11	2.12	∞ : 4.76 : 1
iso-surface animation(sec)	multi	N/A	274	2.12	∞ :129.2 : 1

8. 참고문헌

[1] Andy Cedilnik, Berk Geveci, "Remote Large Data Visualization in ParaView Framework", Eurographics Symposium on Parallel Graphics and Visualization, 2006

[2] COVISE, http://www.hlrs.de/covise

[3] S.Byron, "Virtual Reality in Scientific Visualization", Communications of the ACM, 39(5):62-71, 1996.

[4] Toshiyuki Kimura, "Asynchronous Communication Models for JAX-RPC 2.0", NTT Data Corperation, 2003

[5] Min Ah Kim, "GLOVE(GLObal Virtual reality Environment for scientific simulation): VR환경에서의 대용량 데이터 가시화 시스템", 정보과학회, 2010.

GLOVE Reference Manual

0.01

Generated by Doxygen 1.4.7

Mon Nov 15 14:21:50 2010

Contents

Chapter 1

GLOVE Hierarchical Index

1.1 GLOVE Class Hierarchy

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

Chapter 2

GLOVE Class Index

2.1 GLOVE Class List

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

Chapter 3

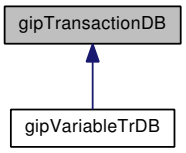
GLOVE Class Documentation

3.1 gipTransactionDB Class Reference

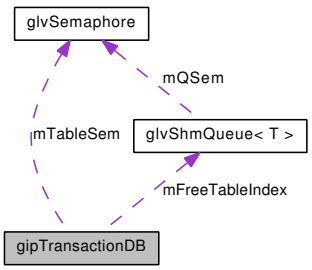
gipTransactionDB (p. ??) is a class for a table with static size records on shared memory.

`#include <gipTransactionDB.h>`

Inheritance diagram for gipTransactionDB:



Collaboration diagram for gipTransactionDB:



Public Member Functions

- **gipTransactionDB (key_t shmKey)**
- **~gipTransactionDB ()**

Destructor.

- **int CreateTable (char *tableName, unsigned int transactionSize, int maxTrNum, int keyPos, int keyLen)**
- **int Insert (void *keyValue, int keyLen, void *tr, unsigned int trLen)**
- **int Insert (void *keyValue, void *tr, unsigned int trLen)**
- **gipTrans * GetTransaction (void *keyValue, int keyLen)**
- **gipTrans * GetTransactionByIndex (int index)**
- **gipTrans * GetUnlockTransaction (void *keyValue, int keyLen)**
- **gipTrans * GetTransaction (void *keyValue)**
- **int Delete (void *keyValue, int keyLen)**
- **int Delete (void *keyValue)**
- **int DeleteByIndex (int index)**

3.1.1 Detailed Description

gipTransactionDB (p. **??**) is a class for a table with static size records on shared memory.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 gipTransactionDB::gipTransactionDB (key_t *shmKey*)

Constructor

Parameters:

shmKey means shared memory key for a table

3.1.3 Member Function Documentation

3.1.3.1 int gipTransactionDB::CreateTable (char * *tableName*, unsigned int *transactionSize*, int *maxTrNum*, int *keyPos*, int *keyLen*)

Create a memory db table

Parameters:

tableName,transactionSize,maxTrNum,keyPos,keyLen

Returns:

success or fail

Reimplemented in **gipVariableTrDB p.** (classgipVariableTrDB_74321afb7b677e97f441d77ca63aef1 **??**)

3.1.3.2 int gipTransactionDB::Delete (void * *keyValue*)

delete a transaction record for an key value when the key position and the key length are already known.

Parameters:

keyValue

Returns:

success or fail

Reimplemented in **gipVariableTrDB p.** (classgipVariableTrDB_c0bf6a5de5c62dd25256b4db41e52b73 ??)

3.1.3.3 int gipTransactionDB::Delete (void * *keyValue*, int *keyLen*)

delete a transaction record for an key value when the key position is already known.

Parameters:

keyValue,keyLen

Returns:

success or fail

Reimplemented in **gipVariableTrDB p.** (classgipVariableTrDB_27ec4eca4a3ef3a95ac62658e0f9f90d ??)

3.1.3.4 int gipTransactionDB::DeleteByIndex (int *index*)

delete a transaction record by the table index

Parameters:

index

Returns:

success or fail

Reimplemented in **gipVariableTrDB p.** (classgipVariableTrDB_044407e1ba67553634d7e5e0971d6f5d ??)

3.1.3.5 gipTrans* gipTransactionDB::GetTransaction (void * *keyValue*)

get a transaction record for an key value when key position and key length are already known.

Returns:

a transaction record data

Reimplemented in **gipVariableTrDB p.** (classgipVariableTrDB$_{fb3ccb94c64eaddf62ff85fc6e2c9746}$??)

3.1.3.6 gipTrans* gipTransactionDB::GetTransaction (void * *keyValue*, int *keyLen*)

get a transaction record for a key

Parameters:

keyValue*,*keyLen

Returns:

a transaction record data

Reimplemented in **gipVariableTrDB p.** (classgipVariableTrDB$_{91d29b5b6d1c219c2c969ff279c15f90}$??)

3.1.3.7 gipTrans* gipTransactionDB::GetTransactionByIndex (int *index*)

get a transaction record for an index

Parameters:

index

Returns:

a transaction record data

3.1.3.8 gipTrans* gipTransactionDB::GetUnlockTransaction (void * *keyValue*, int *keyLen*)

get a transaction record for an index without locking

Parameters:

keyValue*,*keyLen

Returns:

a transaction record data

Reimplemented in **gipVariableTrDB p.** (classgipVariableTrDB$_{e6289a28bf0c6083053b638542f4f690}$??)

3.1.3.9 int gipTransactionDB::Insert (void * *keyValue*, void * *tr*, unsigned int *trLen*)

insert a record for a transaction when key position and key length are already known.

Parameters:

keyValue,tr,trLen

Returns:

success or fail

Reimplemented in **gipVariableTrDB p.** (classgipVariableTrDB$_0$e76208e6238c0ad732872164afab6a0 ??)

3.1.3.10 int gipTransactionDB::Insert (void * *keyValue*, int *keyLen*, void * *tr*, unsigned int *trLen*)

insert a record for a transaction

Parameters:

keyValue,keyLen,tr,trLen

Returns:

success or fail

Reimplemented in **gipVariableTrDB p.** (classgipVariableTrDB$_4$cf0d64a6ef2da47832beaa6a32f5de7 ??)

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

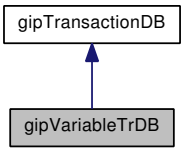
- /home/petimina/glove/trunk/include/trDB/gipTransactionDB.h

3.2 gipVariableTrDB Class Reference

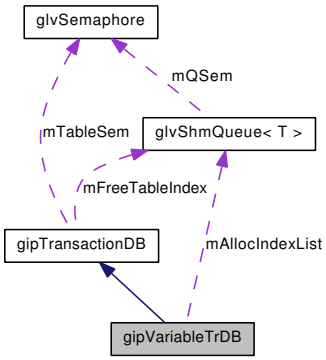
gipVariableTrDB (p. ??) is a class for a table with variable size records on shared memory.

`#include <gipVariableTrDB.h>`

Inheritance diagram for gipVariableTrDB:



Collaboration diagram for gipVariableTrDB:



Public Member Functions

- **gipVariableTrDB (key_t shmKey)**
- **~gipVariableTrDB ()**

  Destructor.

- **int CreateTable (char *tableName, unsigned int transactionSize, int maxTrNum, int keyPos, int keyLen)**
- **int Insert (void *keyValue, int keyLen, void *tr, unsigned int trLen)**
- **int Insert (void *keyValue, void *tr, unsigned int trLen)**
- **gipTrans * GetTransaction (void *keyValue, int keyLen)**
- **gipTrans * GetTransaction (void *keyValue, int keyLen, int keyIndex)**
- **gipTrans * GetUnlockTransaction (void *keyValue, int keyLen)**
- **gipTrans * GetTransaction (void *keyValue)**
- **int Delete (void *keyValue, int keyLen)**

- **int Delete (void *keyValue, int keyLen, int keyIndex)**
- **int Delete (void *keyValue)**
- **int DeleteByIndex (int index)**
- **int FreeIndexInTable (struct gipTrAllocIndexList *indexList)**
- **void Print ()**

  print table contents

- **void Clear ()**

  delete all data in the memory database table

3.2.1 Detailed Description

gipVariableTrDB (p. ??) is a class for a table with variable size records on shared memory.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 gipVariableTrDB::gipVariableTrDB (key_t *shmKey*) [inline]

Constructor

Parameters:

shmKey

3.2.3 Member Function Documentation

3.2.3.1 int gipVariableTrDB::CreateTable (char * *tableName*, unsigned int *transactionSize*, int *maxTrNum*, int *keyPos*, int *keyLen*)

Create a memory db table with **gipTransactionDB** (p. ??) CreateTable

Parameters:

tableName,transactionSize,maxTrNum,keyPos,keyLen

Returns:

success or fail

Reimplemented from **gipTransactionDB** **p.** (classgipTransactionDB$_d$5ba0440b0a9301562657b714a4785c0 ??)

3.2.3.2 int gipVariableTrDB::Delete (void * *keyValue*)

delete a transaction record for an key value when the key position and the key length are already known.

Parameters:

keyValue

Returns:

success or fail

Reimplemented from **gipTransactionDB** p. (classgipTransactionDB$_a$77741dc4674b1cc0db4328253952e44 ??)

3.2.3.3 int gipVariableTrDB::Delete (void * *keyValue*, int *keyLen*, int *keyIndex*)

delete a transaction record for an key value when the key position and the key length are already known.

Parameters:

keyValue,keyLen,keyIndex

Returns:

success or fail

3.2.3.4 int gipVariableTrDB::Delete (void * *keyValue*, int *keyLen*)

delete a transaction record for an key value when the key position is already known.

Parameters:

keyValue,keyLen

Returns:

success or fail

Reimplemented from **gipTransactionDB** p. (classgipTransactionDB$_0$$b0a3f86b04ac6b2c48635e8c75ed$943 ??)

3.2.3.5 int gipVariableTrDB::DeleteByIndex (int *index*)

delete a transaction record by the table index

Parameters:

index

Returns:

success or fail

Reimplemented from **gipTransactionDB** **p.** (classgipTransactionDB$_{dc6c971ebdb9098c912564e62cb76d9c}$??)

3.2.3.6 int gipVariableTrDB::FreeIndexInTable (struct gipTrAllocIndexList * *indexList*)

free a index in the allocated index list.

Parameters:

indexList

Returns:

success or fail

3.2.3.7 gipTrans* gipVariableTrDB::GetTransaction (void * *keyValue*)

get a transaction record for an key value when key position and key length are already known.

Parameters:

keyValue

Returns:

a transaction record data

Reimplemented from **gipTransactionDB** **p.** (classgipTransactionDB$_{cd7099cd781586051bd2681942c7a2b5}$??)

3.2.3.8 gipTrans* gipVariableTrDB::GetTransaction (void * *keyValue*, int *keyLen*, int *keyIndex*)

get a transaction record for an index

Parameters:

keyValue,keyLen,keyIndex

Returns:

a transaction record data

3.2.3.9 gipTrans* gipVariableTrDB::GetTransaction (void * *keyValue*, int *keyLen*)

get a transaction record for a key

Parameters:

keyValue,keyLen

Returns:

a transaction record data

Reimplemented from **gipTransactionDB** p. (classgipTransactionDB$_4$4ff41ba884f5e0ce440ac188a5d3ca1 ??)

3.2.3.10 gipTrans* gipVariableTrDB::GetUnlockTransaction (void * *keyValue*, int *keyLen*)

get a transaction record for an index without locking

Parameters:

keyValue,keyLen

Returns:

a transaction record data

Reimplemented from **gipTransactionDB** p. (classgipTransactionDB$_6$a38756717493af075b83dab405ce346 ??)

3.2.3.11 int gipVariableTrDB::Insert (void * *keyValue*, void * *tr*, unsigned int *trLen*)

insert a record for a transaction when key position and key length are already known.

Parameters:

keyValue,tr,trLen

Returns:

success or fail

Reimplemented from **gipTransactionDB** p. (classgipTransactionDB$_b$767568b86fac25af75113c4e86c6$aab ??)

3.2.3.12 int gipVariableTrDB::Insert (void * *keyValue*, int *keyLen*, void * *tr*, unsigned int *trLen*)

insert a record for a transaction

Parameters:

keyValue,keyLen,tr,trLen

Returns:

success or fail

Reimplemented from **gipTransactionDB** **p.** (classgipTransactionDB$_d$4e9e25c07301793159c7e8313f33ebb ??)

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

- /home/petimina/glove/trunk/include/trDB/gipVariableTrDB.h

3.3 glvSemaphore Class Reference

glvSemaphore (p. **??**) class is a class for supporting linux semaphore

`#include <glvSemaphore.h>`

Public Member Functions

- **glvSemaphore ()**
  Constructor.
- **∼glvSemaphore ()**
  Destructor.
- **int Create (char *semName)**
- **int Lock ()**
  lock the semaphore
- **int Release ()**
  release the semaphore
- **int Close ()**
  close the semaphore

3.3.1 Detailed Description

glvSemaphore (p. **??**) class is a class for supporting linux semaphore

3.3.2 Member Function Documentation

3.3.2.1 int glvSemaphore::Create (char * *semName*)

Create a named semaphore

Parameters:

semName

Returns:

success or fail

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

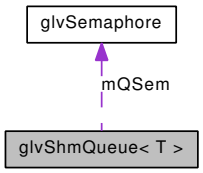
- /home/petimina/glove/trunk/include/util/glvSemaphore.h

3.4 glvShmQueue< T > Class Template Reference

glvShmQueue (p. ??) class is a template class for a queue on shared memory that processes can share.

`#include <glvShmQueue.h>`

Collaboration diagram for glvShmQueue< T >:



Public Member Functions

- **glvShmQueue ()**
  Constructor.
- **~glvShmQueue ()**
  Destructor.
- **int Create (key_t shmKey, char *semName, int qSize)**
- **int Insert (T item)**
- **void Print ()**
  Print all items in the queue.
- **int InsertBackFreeList (int index)**
- **int GetFrontFreeList ()**
- **int Insert (T *itemPtr)**
- **int Insert (T *itemPtr, int num)**
- **int Delete (T *item)**
- **void Delete ()**
  delete a item from the front of the queue
- **int GetFront (T *item)**
- **int GetFrontAndDelete (T *item, int num)**

3.4.1 Detailed Description

template<typename T> class glvShmQueue< T >

glvShmQueue (p. ??) class is a template class for a queue on shared memory that processes can share.

3.4.2 Member Function Documentation

3.4.2.1 template<typename T> int glvShmQueue< T >::Create (key_t *shmKey*, char * *semName*, int *qSize*) [inline]

Create a shared memory queue

Parameters:

shmKey,semName,qSize

Returns:

success or fail

3.4.2.2 template<typename T> int glvShmQueue< T >::Delete (T * *item*) [inline]

delete a item from the queue

Parameters:

item : a item pointer

Returns:

success or fail

3.4.2.3 template<typename T> int glvShmQueue< T >::GetFront (T * *item*) [inline]

Get a front item from the queue without deletion.

Parameters:

item pointer to save a getting item

3.4.2.4 template<typename T> int glvShmQueue< T >::GetFrontAndDelete (T * *item*, int *num*) [inline]

Get front items from the queue withdeletion.

Parameters:

item,num item pointer to save getting items and the number of getting item

3.4.2.5 template<typename T> int glvShmQueue< T >::GetFrontFreeList () [inline]

Get the index of an item from the front of the queue

Returns:

the index

3.4.2.6 template<typename T> int glvShmQueue< T >::Insert (T * *itemPtr*, int *num*) [inline]

Insert a list of items into the queue

Parameters:

itemPtr,num a list of items and the number of items

Returns:

success or fail

3.4.2.7 template<typename T> int glvShmQueue< T >::Insert (T * *itemPtr*) [inline]

Insert a list of items into the queue

Parameters:

itemPtr a list of items

Returns:

success or fail

3.4.2.8 template<typename T> int glvShmQueue< T >::Insert (T *item*) [inline]

Insert a item into the queue

Parameters:

item an item

Returns:

success or fail

3.4.2.9 template<typename T> int glvShmQueue< T >::InsertBackFreeList (int *index*) [inline]

Insert the index of an item into the back of the queue

Parameters:

index

Returns:

success or fail

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

- /home/petimina/glove/trunk/include/util/glvShmQueue.h

3.5 trDBErrCode Class Reference

trDBErrCode (p. **??**) is a class for dealing with error codes for shared memory DB.

`#include <trDBError.h>`

3.5.1 Detailed Description

trDBErrCode (p. **??**) is a class for dealing with error codes for shared memory DB.

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

- /home/petimina/glove/trunk/include/trDB/trDBError.h