Mobile objects and distributed references in open T-System

Konev I. M.

Department of Mathematics and Mechanics
Moscow State University

kim@mail333.com

2005

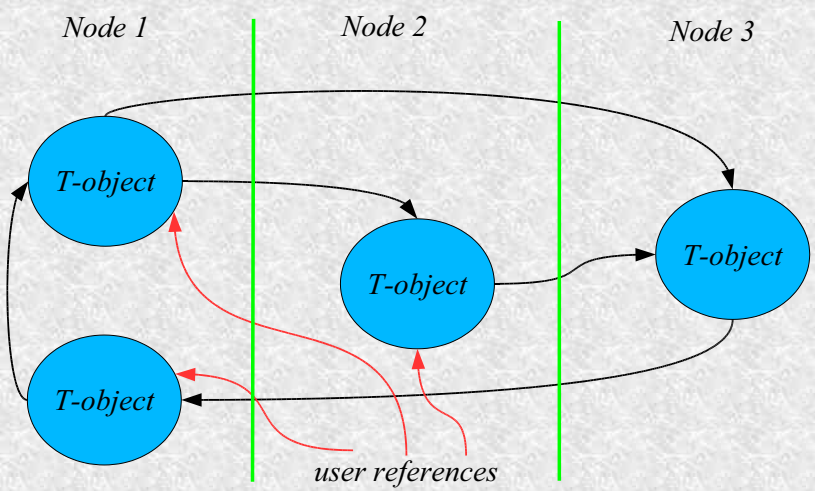
Open T-System – method of parallelizaition of programs

- *TC++ is C++ language enhanced with some new keywords for writing parallel programs*
- *T-variables are special variables that can contain data located on remote node*
- *T-functions are special functions used for distribution of computational work between cluster nodes*

T-variable

- *T-variable is a reference to a special internal object (T-object) that contains data*
- *Different references can point to the same object*
- *Data can contain T-variables inside (references on other T-objects)*

All objects and references to them can be represented as a distributed graph

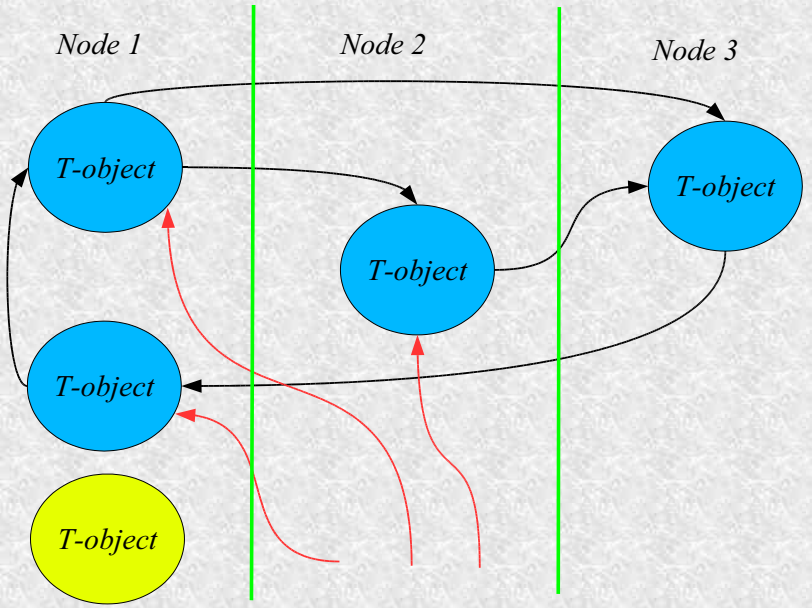


T-objects features

- *All T-objects are created automatically*
- *User gets access to T-objects through dereferencing references*
- *When all the references to T-object are eliminated, it becomes unreachable to user*

As a result, the problem of automatic garbage collection arises: all unreachable T-objects need to be removed.

Main goal of garbage collection



Main components of garbage collection system

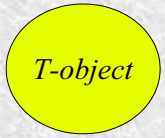
- *Acyclic garbage collector*
- *Local cyclic garbage collector*
- *Global garbage collector*

Speed and functional requirements

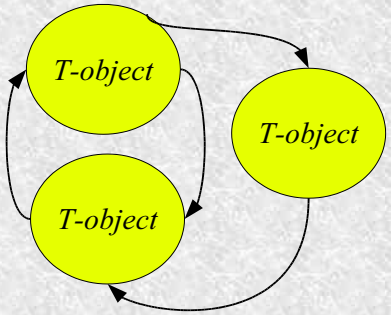
- *Garbage collection system must not slow down user application*
- *All components should be able to work concurrently with user application*
- *Global garbage collector should not need synchronization between cluster nodes*

Acyclic garbage collector (AGC)

AGC removes all objects that have no references to them.



AGC does not eliminate cyclic garbage.



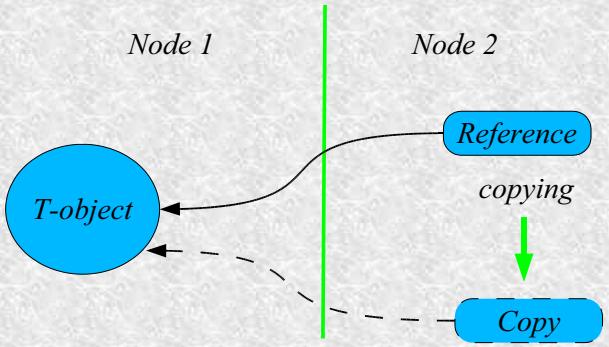
AGC is based on the "reference counting" technique.

Reference counting

Every object stores a positive number in its variable CNT. CNT holds total amount of references, pointed to the object. CNT is changed after any modification of the object.

- *Object creation: CNT = 0*
- *Making a new reference to the object: CNT++*
- *Reference copying: CNT++*
- *Assignment of 2 references, pointing to different objects (A=B): object, corresponding to the reference A – CNT++, object, corresponding to the reference B – CNT--*
- *Deleting the reference: CNT--*

Reference counting in distributed system

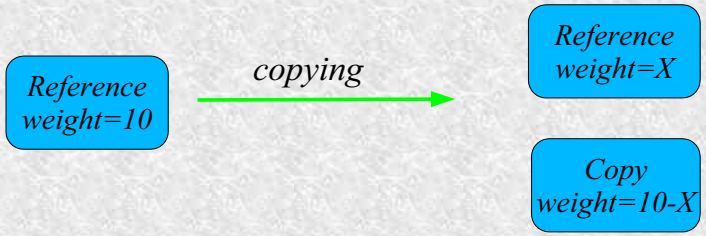


A message (increase CNT) need to be sent after copying the reference on the remote node.

This is unacceptable, because such messages cause essential load of communication channels and extra overhead for message processing.

Weighted reference counting

- *Every reference has certain weight (a positive integer number)*
- *T-object contains total weight, stored in all references*
- *When the reference is copied, its weight is divided in certain way*



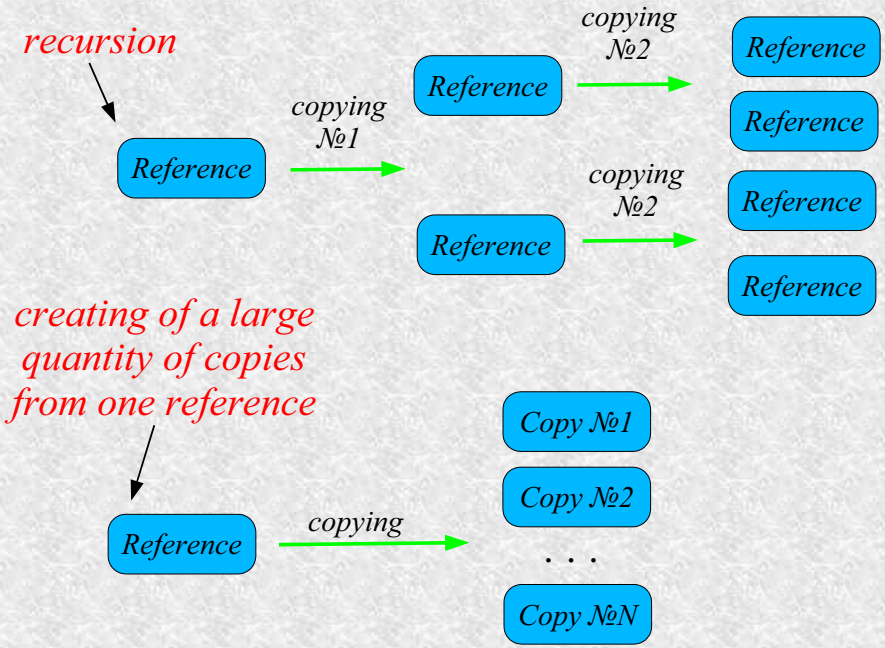
Weighted reference counting: problems

- *What should we do if the weight in the reference can not be divided (weight = 1)?*
- *What ratio is optimal for division of the weight?*
- *How can we avoid sending a message when the reference is eliminated?*

Possible solutions of the first problem

- *Additional object is created. It takes all the weight remaining in the reference. The reference and its copy begin to refer on the just created object.*
- *Request for additional weight is sended to the node containing the object. Functioning of the system is suspended until local node recieves the answer.*

Typical patterns for division of references

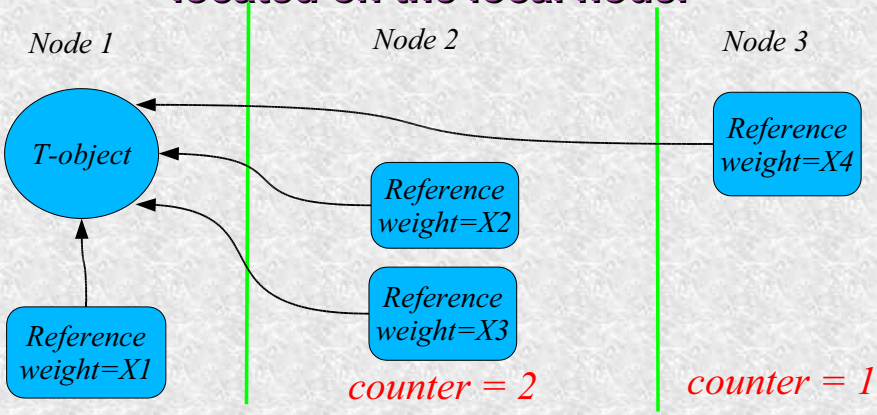


Techniques of the weight dividing when copying the reference

- *Copy takes 1 unit of weight. It does not work with recursive copying.*
- *Copy takes a half of the weight in the reference. It does not work when certain reference is copied many times.*
- *Copy takes sqrt(X), where X is the weight in the reference.*

Deleting the references

Every node stores a special counter for every object, located on the remote node. This counter keeps total number of references pointing to the object, and located on the local node.



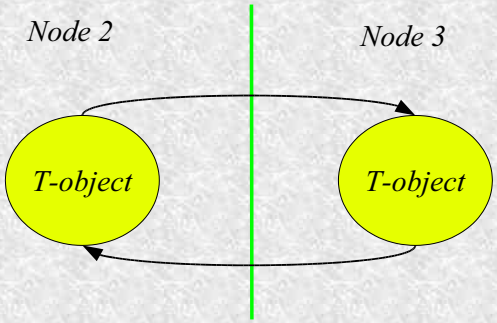
Deleting the references(2)

If a node has N references pointing to the same object on the remote node, and deletes them one by one, the message with the weight will be sent only once.

As a result, overhead caused by sending N-1 extra messages is totally eliminated.

Local cyclic garbage collector (LGC)

LGC does not allow to collect cyclic garbage, distributed among several nodes.



LGC collects all garbage, located on a single node.

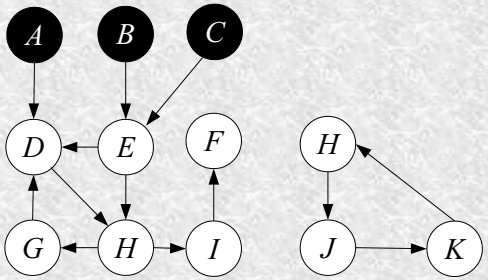
Local cyclic garbage collector(2)

- *The LGC is based on "mark-and-sweep" technique. Basic mark-and-sweep collection was adapted to the features of T-system.*
- *LGC requires cooperation with acyclic garbage collector.*

Mark-and-sweep collection

- *Mark-phase. All objects, accessible to user, are marked.*
- *Sweep-phase. All inaccessible to user objects are eliminated.*

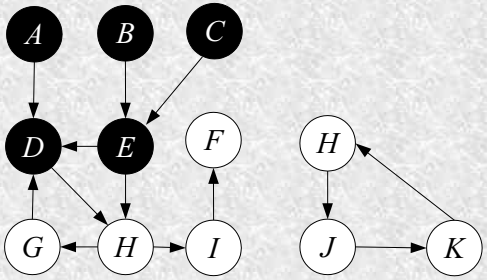
At first, a set of objects, accessible by user in any case (so-called root set), is determined. Then all the objects from the root set are marked.



Mark-and-sweep collection(2)

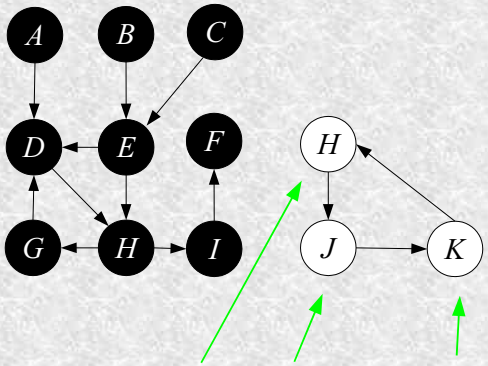
Futher, all the references, contained in marked objects, are examined; objects, corresponding to these references are marked. This process lasts until all marked objects contains references to the marked objects.

State after first pass.



Mark-and-sweep collection(3)

Final state.



All unmarked objects need to be eliminated.

Mark-and-sweep collection in distributed case

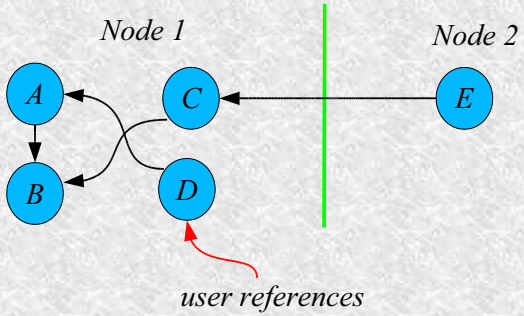
- *Determining the root set causes big truoble, because the objects located on remote nodes can contain references to the objects on the local node.*

Two ways to determine root set

- *Every node keeps a list of objects, accessible by reference from remote nodes. The list is updated when reference is deleted or moved.*
- *Root set is determined by means of information necessary for acyclic garbage collector.*

Determining the root set

Total weight of the local references is calculated for every object, located on a node. If this computed value is not equal to the weight in the object, then the object belongs to the root set.



Local cyclic garbage collector: stages

- *Determining the root set with the help of information about weight*
- *Mark-phase: marking all the objects accessible to user*
- *Sweep-phase: eliminating all unmarked objects*

Global garbage collector (GGC)

- *GGC is based on a distributed mark-and-sweep collection algorithm*
- *It collects all garbage*
- *Does not require synchronization between nodes*
- *May work concurrently with user application*

Global garbage collecor: stages

- *Preparation: determining the set of the objects for distributed mark-and-sweep*
- *Determining the root set*
- *Mark-phase*
- *Sweep-phase*

Determining the set for mark-and-sweep

1. *Fixed node (node X) sends a message about the beginning of the first stage to all the rest nodes.*
2. *When certain node receives the message, it stores the state of all local objects in special repository. After that node sends the message about the finish of the first stage back.*
3. *Node X waits until the rest nodes reply.*
4. *Then node X sends a message about the beginning of the second phase to the rest nodes.*

Determining the set for mark-and-sweep (2)

5. *Every node gets a message about the beginning of the second phase. Then node checks if the objects were changed after finishing the first phase. All changed objects are removed from repository. After that node sends the message about the finishing of the second phase back.*
6. *Node X waits until the rest nodes reply.*
7. *After that the GGC begins a new collection cycle within the objects, remaining in repository.*