Expresso: a library for fast transfers of Java objects

Yves Mahéo, Luc Courtrai, Frédéric Raimbault

Orcade Project

Valoria, Université de Bretagne Sud,France

Motivation and objectives

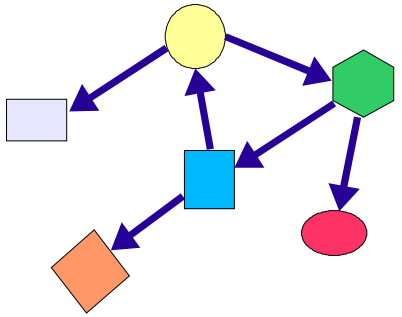
- Motivation
  - ➤ Distributed programming with objects needed for high−performance platforms
  - ➤ Current runtimes assume low performance networks
  - ➔ Study of the impact of high performance networks on runtime systems for distributed object oriented environments
  - ➔ Proposal for new implementation techniques
- First work
  - ➤ Provide object transfer that exploits high perf networks
  - ➤ Prototype library

Context

- Object transfer between two nodes
  - Java
  - Homogeneous local area network
    - same architecture

      same OS

      Myrinet network
  - Graph of objects with sharing and cycles
  - Transfer = copy

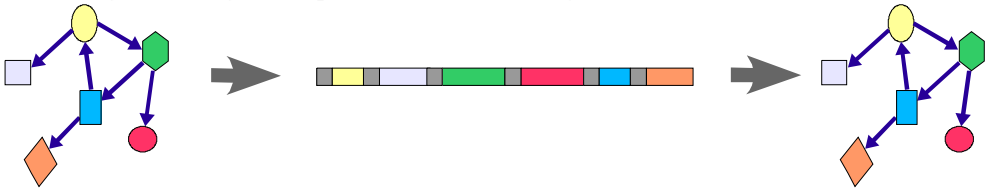


Serialization

- Usual method for transferring a graph of objects
  - graph traversal

    management of cycles

    message = data + type information



  - Java *Serializable* Interface

    automatic

    very general

    not efficient (introspection, non optimized implementation...)

  - Java *Externalizable* Interface

    specialization by the programmer

  - Research on optimization

    native serialization code, automatic compilation

    pure Java solutions with optimized buffer management, reduced message size...
    (XSerializable interface by the JavaParty team)

Serialization performances

- Transfer time measurements
  - ➤ serialization + communication + unserialization
  - ➤ binary tree with 1000 nodes

	Serializable	Externalizable	XSerializable
Total Ethernet	**148 ms**	**31 ms**	**23 ms**
Part of ser/unser	**73 %**	**37 %**	**16 %**
Total Myrinet	**109 ms**	**12 ms**	**4 ms**
Part of ser/unser	**99 %**	**95 %**	**86 %**

➔ Question the serialization principle

Expresso

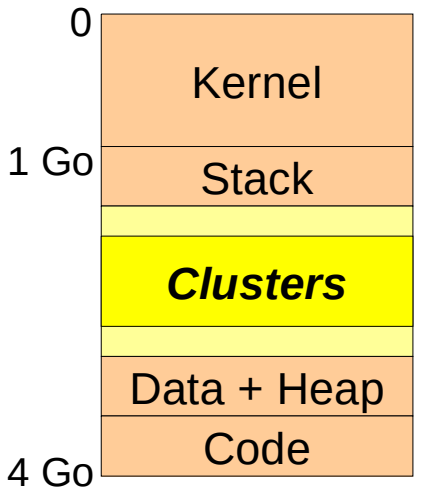
- Library for object transfers
- Java package
- Characteristics :
  - clustering of objects
  - memory allocation control
  - Iso−address transfer
  - no new distributed model
    - based on MPI :

      1 process per node

      blocking send, non blocking receive

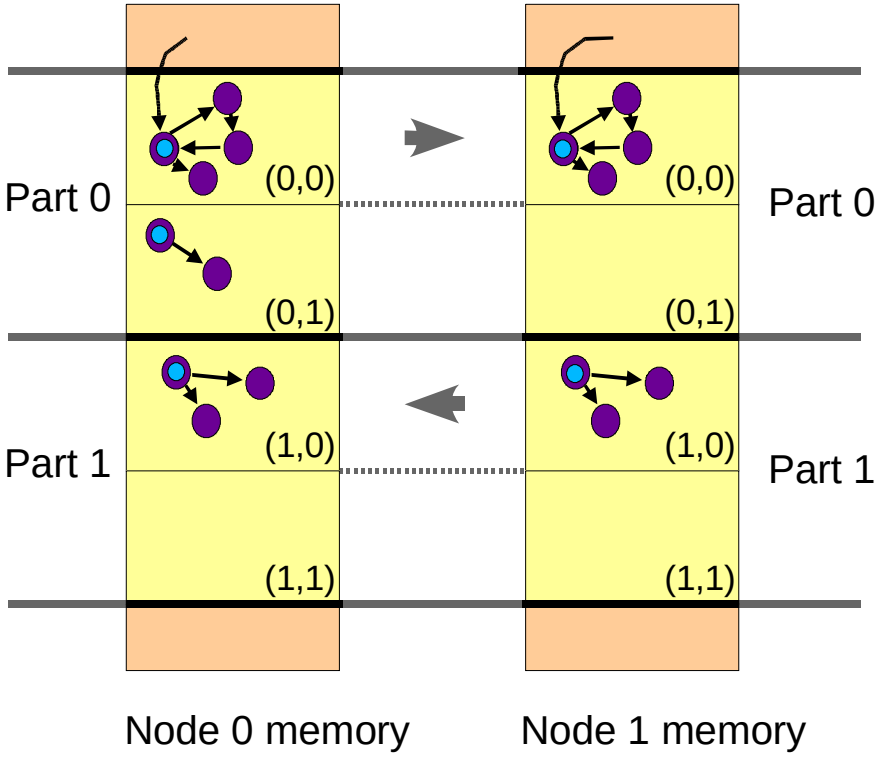
Iso−address transfer

- Principle : place the transferable objects at the same address on the receiver as on the sender
  - References remains valid after transfer
  - No extra updating on the receiving side
  - Efficient graph transfer = transfer of a memory block
- Reserved range of virtual addresses on each process
  - Only transferable objects are allocated this way
  - Others objects allocated in the JVM heap



Objects Clustering

- Cluster = communication unit
  - Contains one graph of objects
  - Root object
  - Global naming (creating node, rank)
    - placement, number and size fixed at initialization time
  - No inter−clusters references



Expresso API

- Java Package
  - ClusterISO class
- Typical use
  - Sender :
    - create a cluster object

      setting it as the current cluster

      create objects that forms a graph

      send the cluster to another node
  - Receiver :

    receive the cluster

    access the objects

Example

```
ClusterISO.init(argv);
if (ClusterISO.myNode == NODE0) {
    ClusterISO clu = new ClusterISO(3);
    clu.setCurrent();
    Btree mytr = (Btree)ClusterISO.newObject(Btree.class);
    mytr.lc = (Btree)ClusterISO.newObject(Btree.class);
    mytr.rc = (Btree)ClusterISO.newObject(Btree.class);
    ...
    clu.setRootObject(mytr);
    clu.send(NODE1);
}
else {
    ClusterISO clu = new ClusterISO(3);
    clu = ClusterISO.recv(NODE0,3);
    mytr Btree = (Btree)clu.getRootObject();
    mytr.print();
}
ClusterISO.quit();
```

Implementation

- Based upon
  - Kaffe JVM (1.0.5)
    - Written in C (Java Native Interface)
  - Myrinet

    MPI library (gm−mpich 1.2..3)
- Two parts
  - Memory management
  - Communication management

Memory management

- Initialization
  - Iso−address space allocation
    - fixed sized Iso−clusters

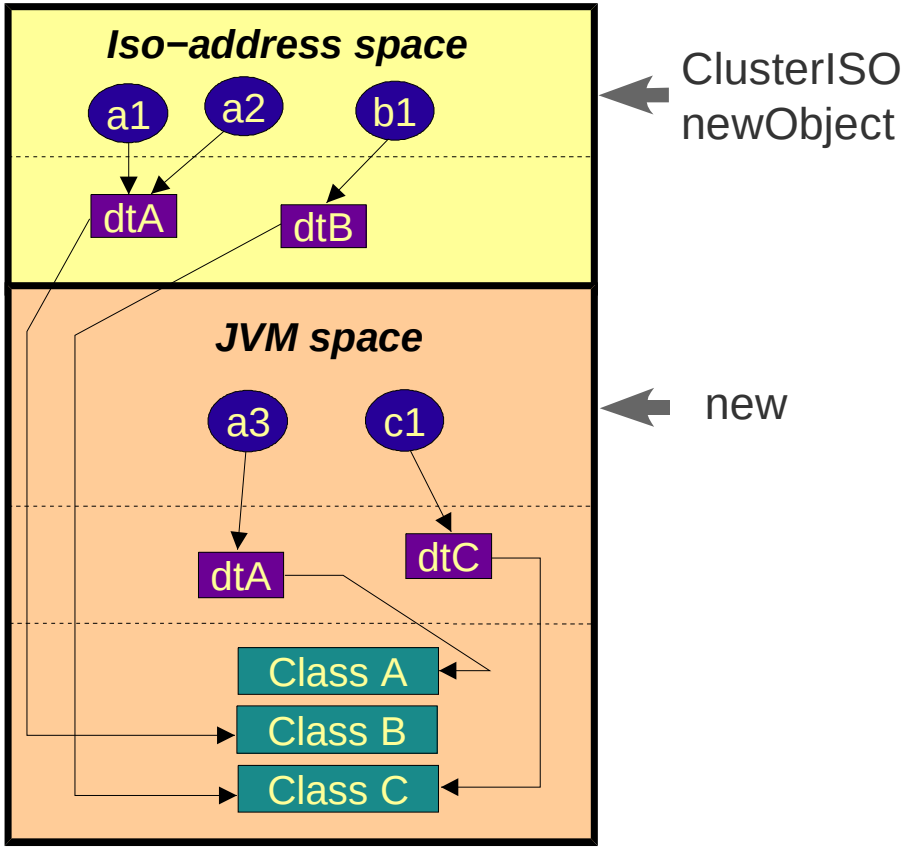
      copies of dispatch tables

    pre−loading of classes
- Object creation

  NewObject method
  for transferable objects

  JVM new for other objects

  same structure for both types of objects



Communication management

- Instance methods from class ClusterISO
  - Java wrappers of C functions
  - Dynamic library build with MPI modules
  - gm−mpich 1.2..3 over GM 1.01
- Graph transfer = direct transfer of a memory block
  - only one MPI message
  - zero copy
  - no extra computation on the sender and on the receiver

Performances

- Creation and access
  - ➤ newObject faster than JVM new
    - ◆ more efficient access to class data
  - ➤ time for accessing objects : identical
- Transfers
  - ➤ measurement on a random graph (road map)

    3 types of objects

    4 references per object (average of 2.5 non null)

    small size of data

    ➔ message 3 times longer than the serialized form of the graph

Transfer times

- Platform
  - PC Pentium II 4OO Mhz, Linux 2.036
  - Myrinet PCI 33 Mhz, Lanai 4, 8−port switch
- Graph transfer times in *ms*

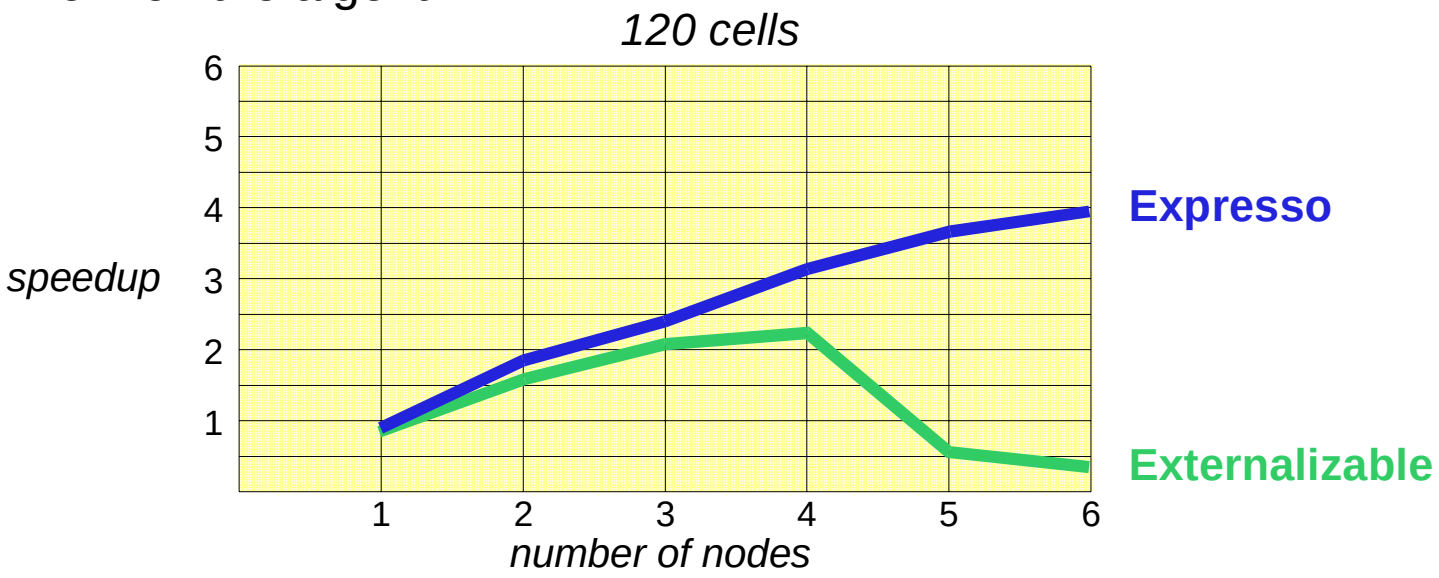
# Towns	# Objects	Serializable	Externalizable	ClusterISO
10	**29**	**25**	**3**	**0.28**
100	**355**	**145**	**25**	**0.81**
500	**1760**	**1384**	**210**	**2.82**

Application

- Genetic Programming
  - Parallelization of a Java sequential code
    - simple parallelization scheme
      - data distribution by block

        computation + scatter−gather phases

      no particular work on the algorithm



Conclusion

- Expresso
  - ➤ Clustering + Iso−address transfer
  - ➤ Avoid costly serialization
  - ➤ Prototype
  - ➤ High performance
    - ◆ Fully exploits the network capabilities

      Can be considered as an upper bound
- Improvements
  - ➤ Inter−clusters references
  - ➤ Transfer of objects created by new
  - ➔ Modification of the JVM