The Ka tools

and

OSCAR

Simon Derr, INRIA

Simon.Derr@imag.fr

January 2002

Goals of this presentation

- Integrate some ideas of Ka in OSCAR
- Establish a collaboration between INRIA and OSCAR

Who are we ?

- INRIA : institut national de recherche en informatique et automatismes

French public institute that does research in computer science

- the APACHE project
  - City of Grenoble

- Fundings from MS, BULL for previous works
- Fundings from the French Govt for a “cluster oriented Linux distribution” in association with Mandrake.

ID-Apache

Objectives : Distributed computing

- Cluster of multiprocessors (CLUMP) for CPU intensive applications
- Performance, “easy access”, scalability, heterogeneity and resilience

Research directions

1) Parallel programming model
2) Scheduling and load balancing
3) Management tools
4) Parallel algorithms

Validation

1) A parallel programming environment Athapascan
2) For real applications
3) On significant parallel platforms (few hundreds to thousands)

Interest in clusters of PC's

- One-year old cluster of 225 uniprocessors PIII
  - 100 mbit fast ethernet
- Process of buying a more powerful machine
  - Around 128 dual-processor nodes
  - High performance network

Ka tools

- Scalable tools
  - Designed to fulfill the needs we had on our 225-node fast-ethernet cluster
  - Ka-deploy
    - OS installations
  - Ka-run
    - launching of parallel programs, run commands on the cluster
    - files distribution

- And also...
  - Monitoring
  - Distributed NFS

Idea behind Ka

- 2 goals
  - Contact many nodes from one node (contact = run a remote command)
  - Send big amounts of data to many nodes from one node
    - On our ‘slow’ switched fast-ethernet network

- Problem : source node bottleneck
- One common solution : trees

Using trees to run a command

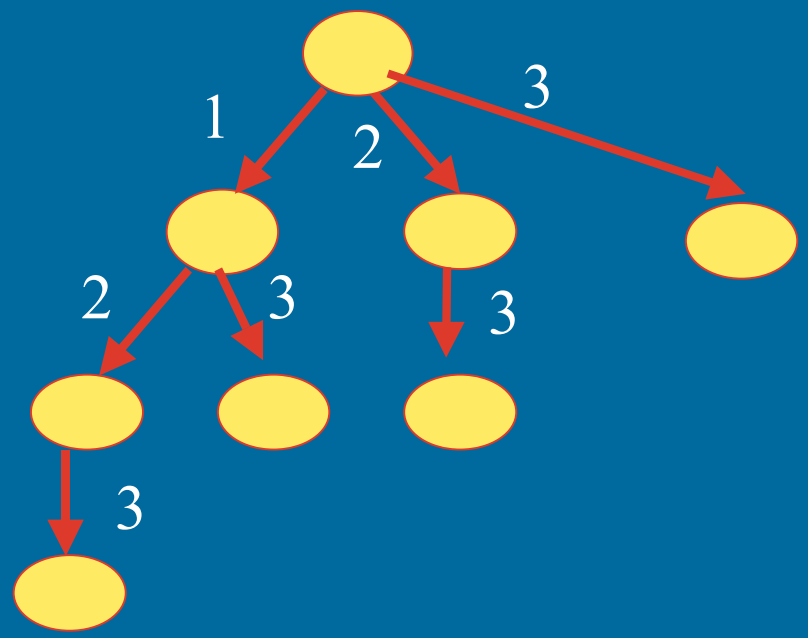
Objective : quickly contact many nodes (contact = rsh)

Contacting many nodes from a single host produces a lot of network traffic and cpu work

Idea: contact a few nodes and then delegate some of the work to the nodes that have already been contacted == use a tree

ex: binomial

Using trees to run a command



Implementation : *rshp*

Comparison with C3

- Running commands with C3 *cexec*
  - All nodes contacted by a single node
    - Network traffic
  - A process forked() for each destination node -> high cpu load on the source node
- Running commands with rshp-enabled *cexec*
  - Each node contacts only a few other nodes
  - No per node fork() (when rsh -not ssh- is used)
  - Tree brings scalability

Comparison with C3

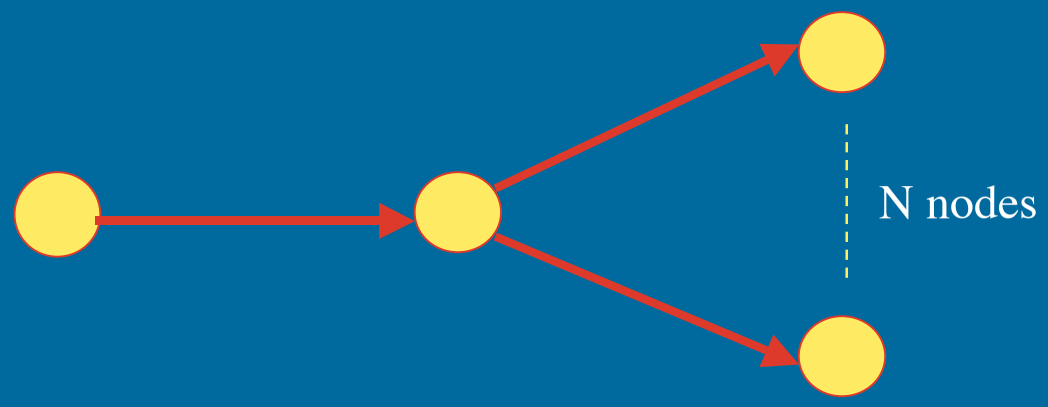
Time to run the *uname* command on 130 machines of our cluster:

- Time with *cexec*: `0:02.07 elapsed 85%CPU`
- Time with *rshp-enabled cexec* : `0:01.50 elapsed 8%CPU`
  - Using a binomial tree
  - Future : Non-blocking connect() calls to improve speed

Using trees to send data

Objective : high bandwidth

Idea : create a structure of TCP connections that will be used to send the data to all the machines



On a SWITCHED ethernet-like network:

One node receiving data and repeating them to N other nodes

Bandwidth = network bandwidth / N

Using trees to send data

Objective : high bandwidth

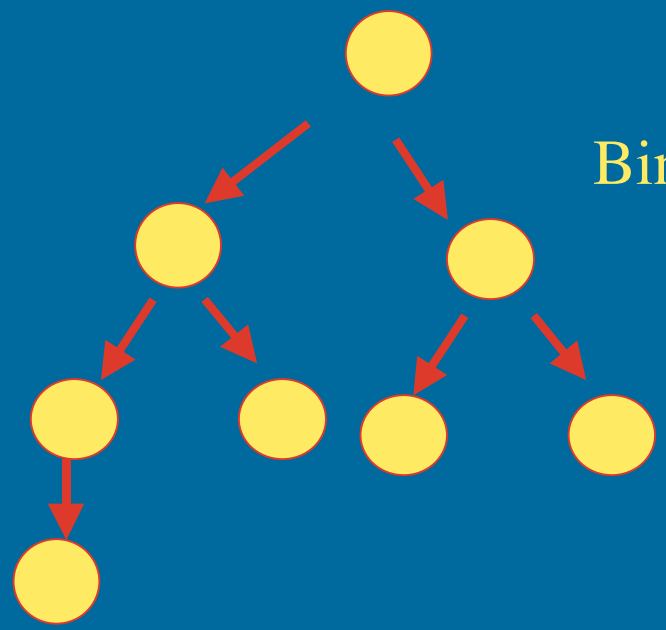
Idea : create a structure of TCP connections that will be used to send the data to all the machines

On a switched network where the bandwidth limitation is *per host* and where there is no limitation on the total traffic on the network. *where* the bottlenecks are the hosts and not the switchs…

Consider a node on a fast ethernet network (full duplex). If it receives data on one side, and sends it to *k* nodes on the other, it can do so at the speed of 100/k mbit/s (in real world, ~about 10/k MB/s)

+pitit schema de 1 noeud

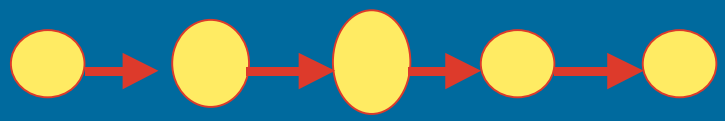
Using trees to send data



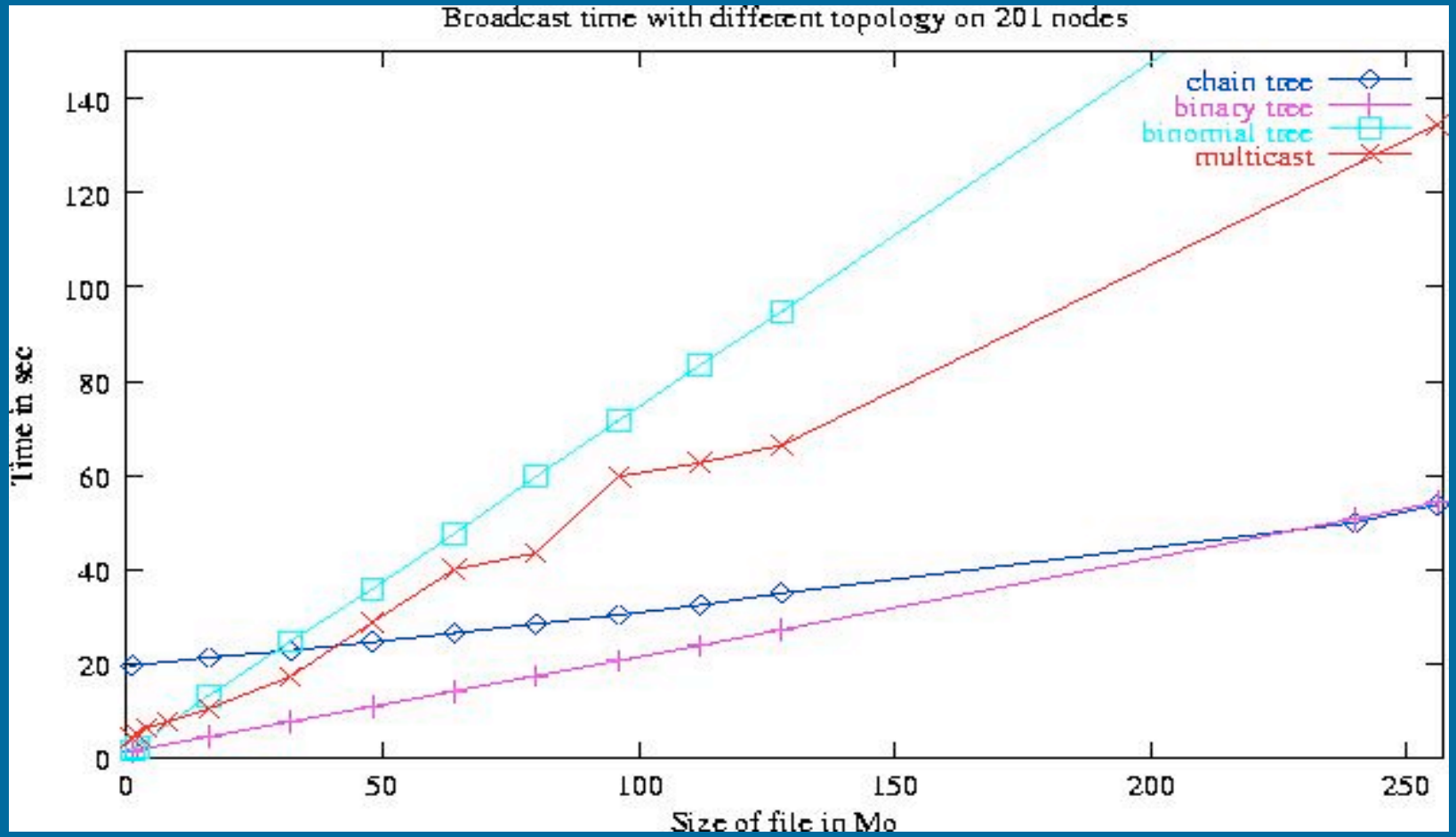
Binary tree on a fast ethernet network : ~ 5 MB/s

Chain tree on a fast ethernet network : ~ 10 MB/s

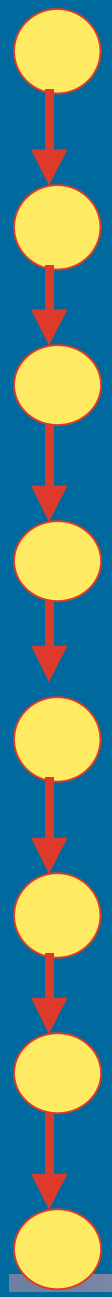
BUT tree creation takes longer (very deep tree)

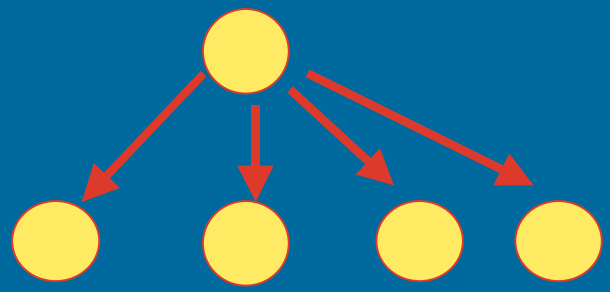


File transfer

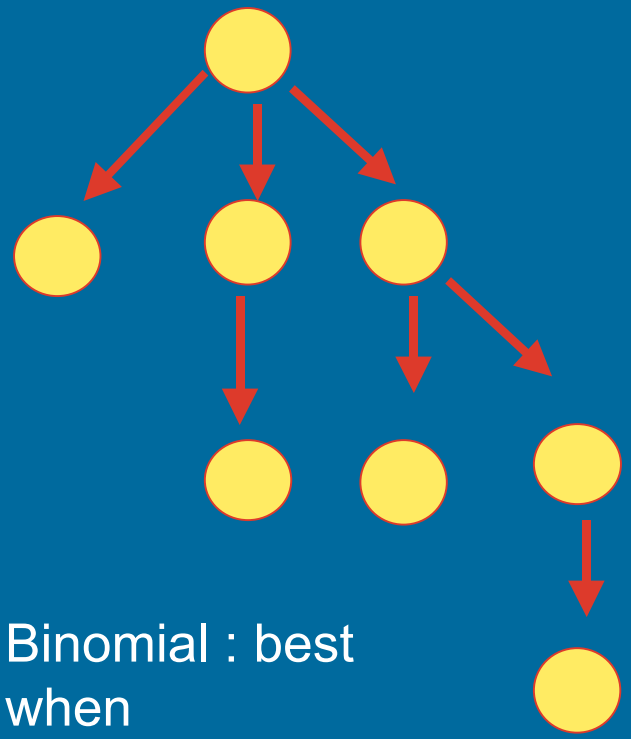


Topologies and their use

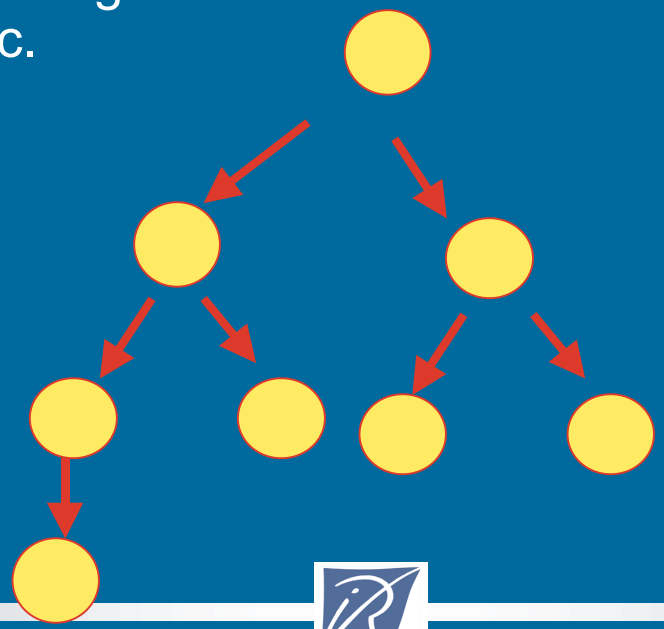




Sequential : best when asynchronous until ….eg rsh async.



Binomial : best when synchronous, eg rsh



Chain : best when start-up is small, eg file transfert

Binary : better for subsequent comm., eg monitoring

Comparison with C3

- Sending files with C3 *cpush*
  - Use of rsync : efficient for modified files
  - Sending new files *(blind mode):*
    - Network bottleneck on the sending node
    - Transfer time linear / number of nodes
- Sending files with rshp-enabled cpush
  - rshp duplicates stdin : sending a file is merely :

    `cat filein | rshp options dd of=fileout`
  - Transfer time *almost* independent / number of nodes

Comparison with C3

Time to send a 30MB file to 20 nodes:

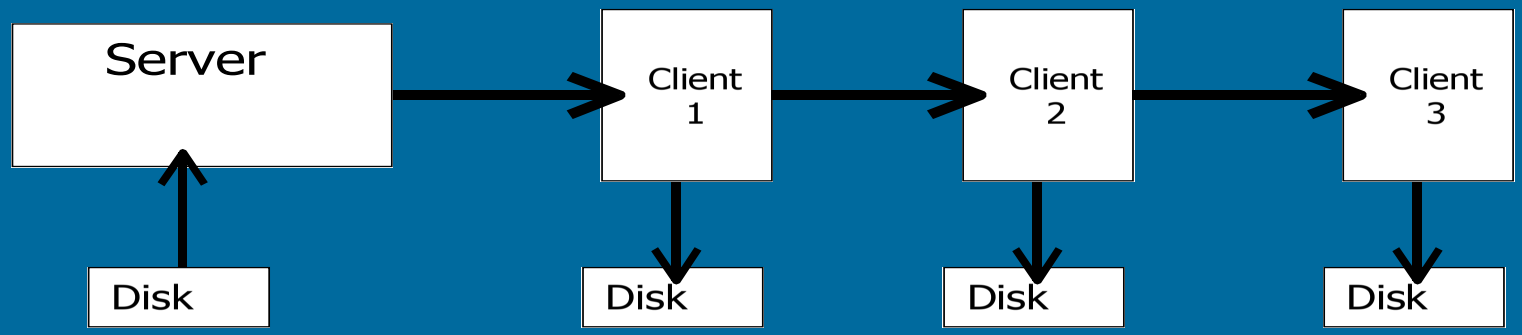
- Time with *cpush*: `1:12.67 elapsed 99%CPU`
- Time with *rshp-enabled cpush* : `0:05.88 elapsed 21%CPU`

Possible integration with C3

- Current C3 code handles inter-cluster stuff, reads the cluster description files, parses the command line, …
- Rshp only handles and accelerates intra-cluster communications for *cexec*, and intra-cluster data transmission in *cpush*'s *blind* mode.
  - For now only if C3_RSH is 'rsh'
  - Next version of *rshp* should be able to use ssh

Ka-deploy

- Scalable operating system installation (*almost*)
- Node duplication
- PXE-capable cluster nodes network-boot and use a TCP chain-tree to efficiently transfer OS files



- Works on Linux, for Linux and Windows

Ka-deploy

- Speed : installation of a 1-2 GB system on 200 machines can take less than 15 minutes
- Very little flexibity
  - Machines must be homogenous
  - Very painful to set up

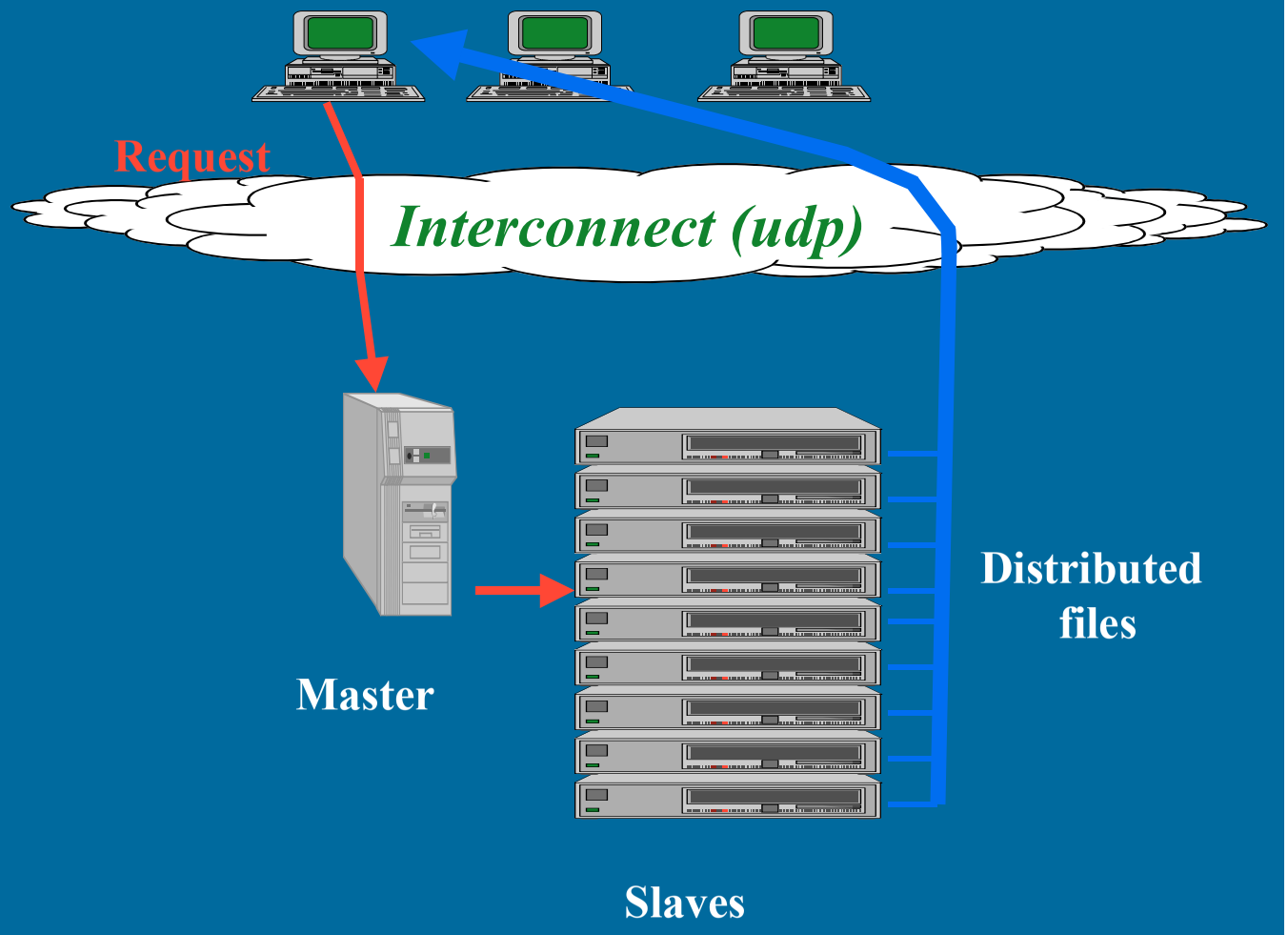
Ka-deploy and LUI

- Same environment : PXE boot, etc…
- Different goals:
  - LUI is headed towards flexibility, and ease of use
  - Ka-deploy is headed towards speed and scalability
- Maybe the diffusion scheme used by ka-deploy can be added in LUI
  - But with SIS ??

NFS server for clusters

The cluster is the file system

- **NFS client unchanged**
- **files placement**
  - **parallel access**
- **scalability ??**
  - **optimized read**
  - **write??**



Conclusion

- Very interested in a collaboration
  - Some manpower, and one (soon 2) clusters for testing
  - Visitors are welcome
  - Maybe even host a future meeting

- Other research directions:
  - Peer to peer machine cloning
  - Intranet clusters

Web : icluster.imag.fr, ka-tools.sourceforge.net