Optimizing Metacomputing with Communication-Computation Overlap

Françoise Baude, Denis Caromel, Nathalie Furmento, David Sagnol

OASIS project: INRIA Sophia-Antipolis and I3S (CNRS-Univ. of Nice, France)

Email: `FirstName.LastName@sophia.inria.fr`

6th PaCT conference — September 2001

http://www.inria.fr/oasis/

Context - General Objective

- Improve runtime performances of distributed applications by overlapping communication with computation.

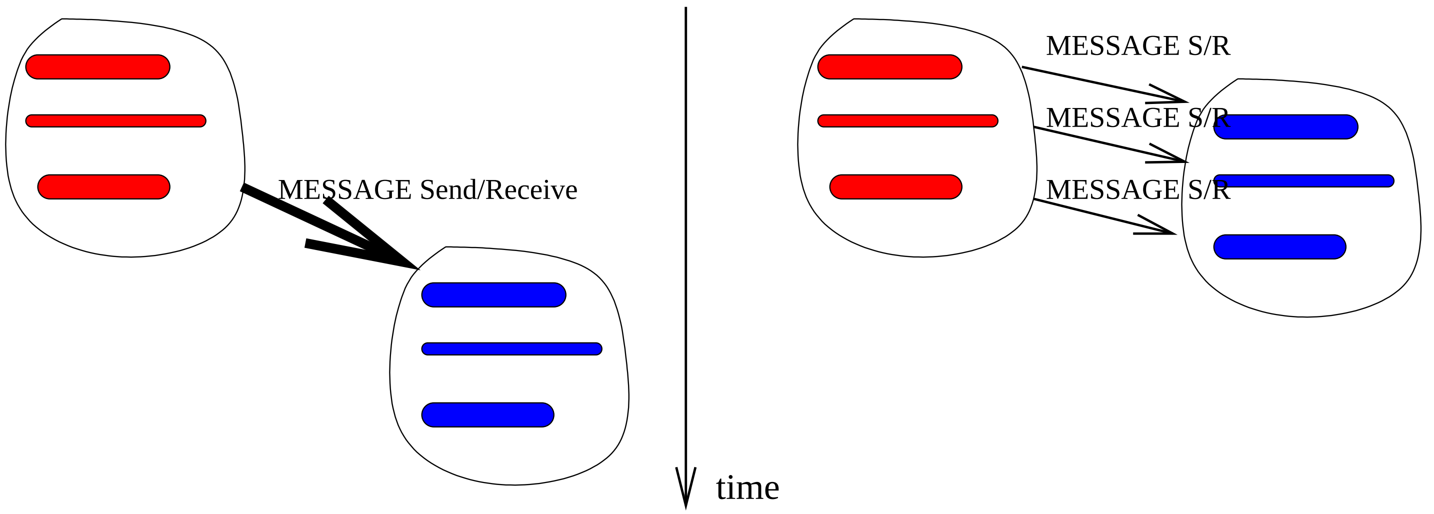
- Not a new idea (cf. parallel & distr. computing) but metacomputing raises new assumptions
  - Applications run on Grids (i.e. interconnected heterogeneous platforms)
  - Grids are based on low performance communication medium (sometimes WANs)

$\Longrightarrow$ Improve communication performances

Formulation of the overlapping problem (1/2)

- Split data to send into many pieces
- Send each piece in turn
- Compute with a piece as soon as it is received

$\Longrightarrow$ Requires asynchronous communication primitives



⟹ Pipeline effect between elementary send/receive and compute operations

⟹ In theory, all the duration related to sending pieces of data (except the first one) can be spared

Formulation of the overlapping problem (2/2)

- How to program the overlapping (usually in a PVM/MPI message-passing style) ?
  - By hand: program yourself the splitting, sending, reception, computation $\Longrightarrow$ tedious, error-prone, not transparent, no good reuse of the previous code
  - The library: by assuming communication and computations performances are known in advance, the whole distributed algorithm is redesigned and implemented as a new routine in the library (ex. in LOCCS) $\Longrightarrow$ not universally applicable

This work: Apply overlapping to distributed Object-Oriented systems

Claim: such O.O systems help decrease the programming efforts

- Improve runtime performances of Object Oriented distributed and metacomputing applications based on remote services by

  OVERLAPPING

  the transmission of remote service request

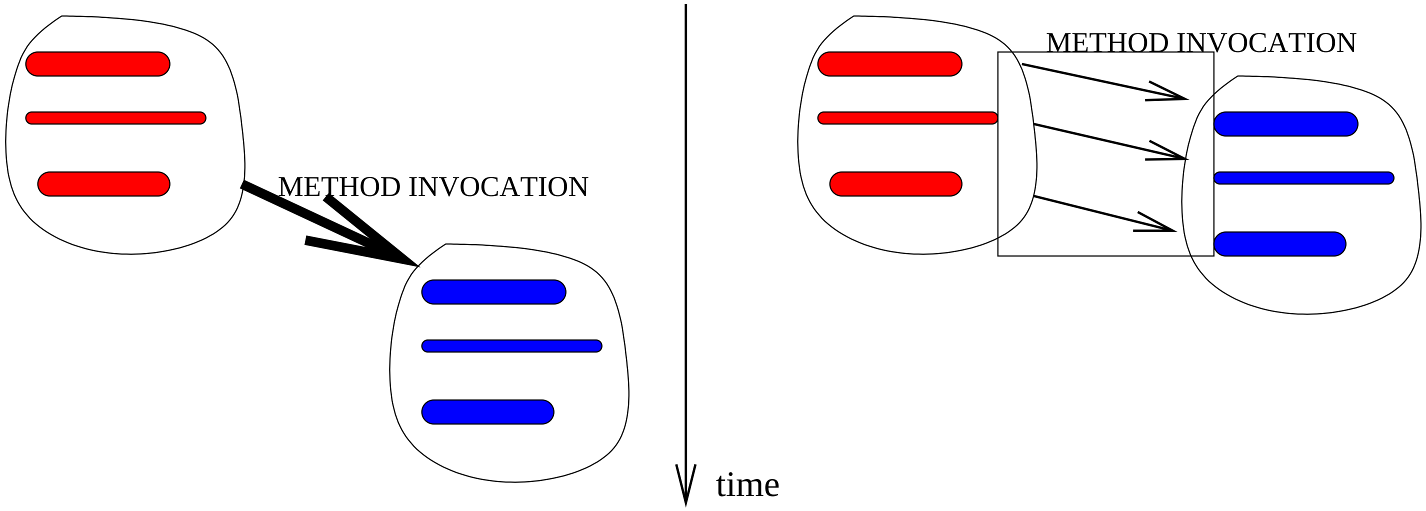
  with

  the remote execution of this request

- As most as possible transparent for the programmers.

Strategy

Be able to start a remote service before receiving the whole set of service request arguments.



⟹ Modify the transmission/execution mechanisms of the request in the runtime system only with some minimal help from the user code.

CONTENTS

How To Obtain Overlapping ?

- Request Creation and Transmission:
  - Dissociate the splitting/marshalling operations from the sending ones. Be able to decide when to send a request piece.
- Request Rebuilding:
  - Identify the missing data. Transparently update them when they come.
- Service Execution:
  - Automatically blocked when data are missing. Resumed when OK.

Where to Split a Request ?

- Basic solution: splitting done by the runtime system. E.g.: while traversing the graph of objects to be marshalled, each branch becomes a new request piece.

- Modification of Class: overrides the default marshalling routine. E.g: each line of a matrix becomes a new piece

- In practice, the splitting is better *guided* by the programmer:
  - Indicate classes that have to be sent later (through a specific inheritance). Possible to reverse the indication (through a cast)
  - The generic marshalling/unmarshalling functions are modified to take into account these indications.

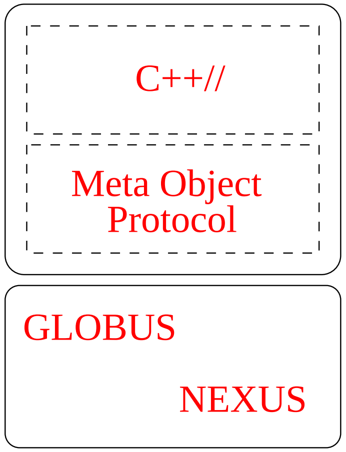
CONTENTS

C++//, a Distributed Language (Library)

- Heterogeneous Model with both Active and Passive Objects,
- Asynchronous Communications towards Active Objects,
- Wait-by-necessity (Automatic and Transparent Futures),
- Generic marshalling and unmarshalling routines,
- Same model is used in ProActive PDC (Java//)

C++// Runtime Support

- Based on Nexus/Globus: a toolkit to build distributed and metacomputing applications.
  - Each C++// active object is implemented by a Nexus thread
  - Method invocation towards a C++// active object is implemented by a Nexus remote service request.
  - Deployment of active objects on metacomputing platforms is done with Globus



Implementation of the Overlapping

- in the C++// library
  - Objects (arguments of requests) to be sent later are of C++// Future type.
- in the C++// runtime system on top of NEXUS
  - Each request piece is transmitted via a new Nexus remote service request ($\rightarrow$ transparently received and treated).
  - Redefinition of the `send` method between clients and servers such as to send a request through several transmissions.

$\Longrightarrow$ very few modifications

CONTENTS

✓ • Introduction

✓ • Design of the Overlapping Mechanism in O.O. Systems

✓ • Prototype Environment

☞ • Performances

  - Example
  - One Experiment onto a LAN
  - Benefit: Is the real gain close to optimal ?
  - Benefit of the LAN-based experiment
  - One Experiment onto a WAN-based Grid

– . . . and its positive and close to optimal gain

- Conclusion

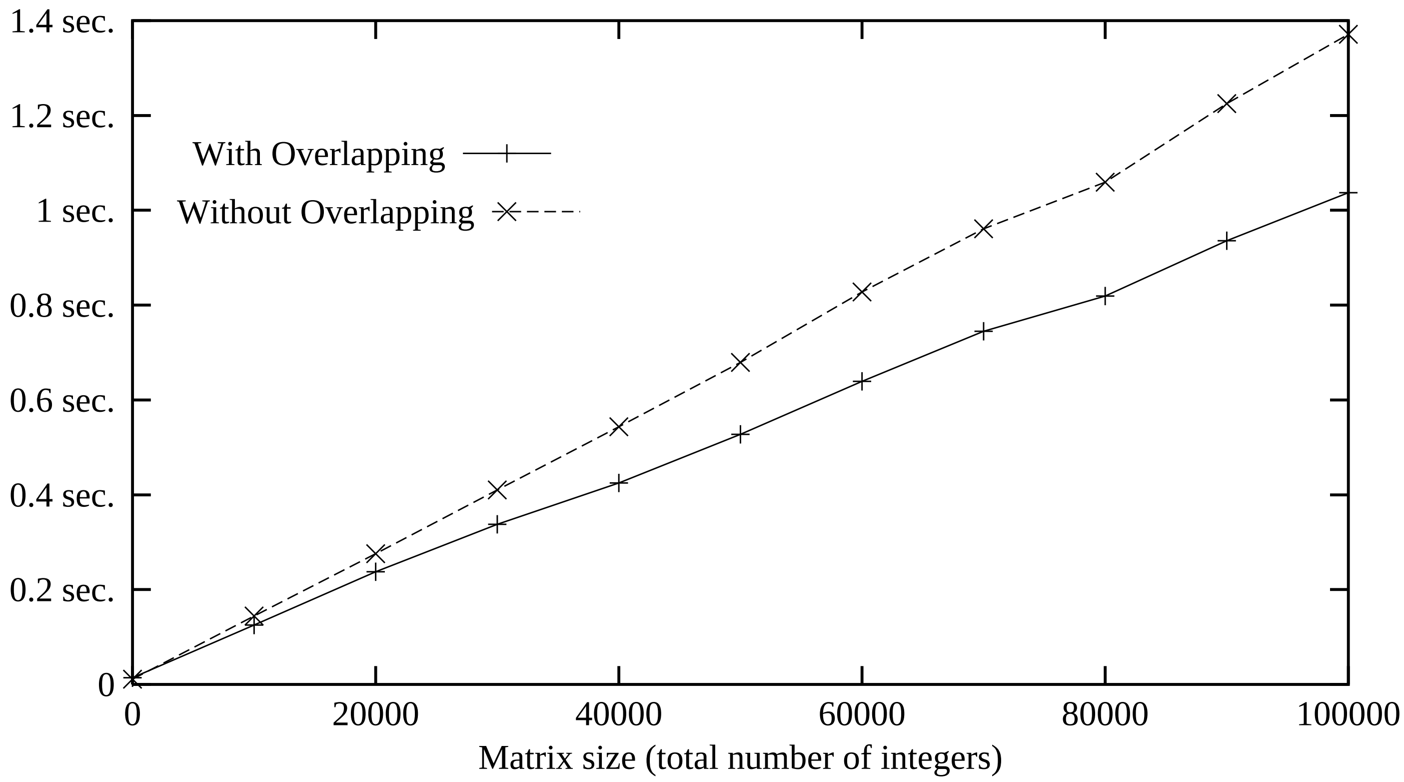
Example

```
class OpMatrix : public Process {
	virtual int rank (Matrix *m1, Matrix *m2) {
		m1->square();
		m2->plus(m1);
		int res = m2->result();
		return (res);
	}
};
class Matrix_Future :
	public Future, public Matrix {
...
}
```

```
OpMatrix *dom = CppLL_new(
	("myhost"),OpMatrix,());
Matrix *m1 = new
	Matrix(COLUMN, LINE);
Matrix *m2 = CppLL_new (
	Matrix_Future, (COLUMN, LINE));
// set the values for m1 and m2
CLOCK_Call_Time_START;
int res = dom->rank(m1, m2);
CLOCK_Call_Time_STOP;
```

$\Longrightarrow$ Ultimate goal: spare the transmission duration of the Matrix m2 in the remote rank service execution.

One Experiment onto a LAN



Gain shown for one rank method call; Scales well (multithreading)

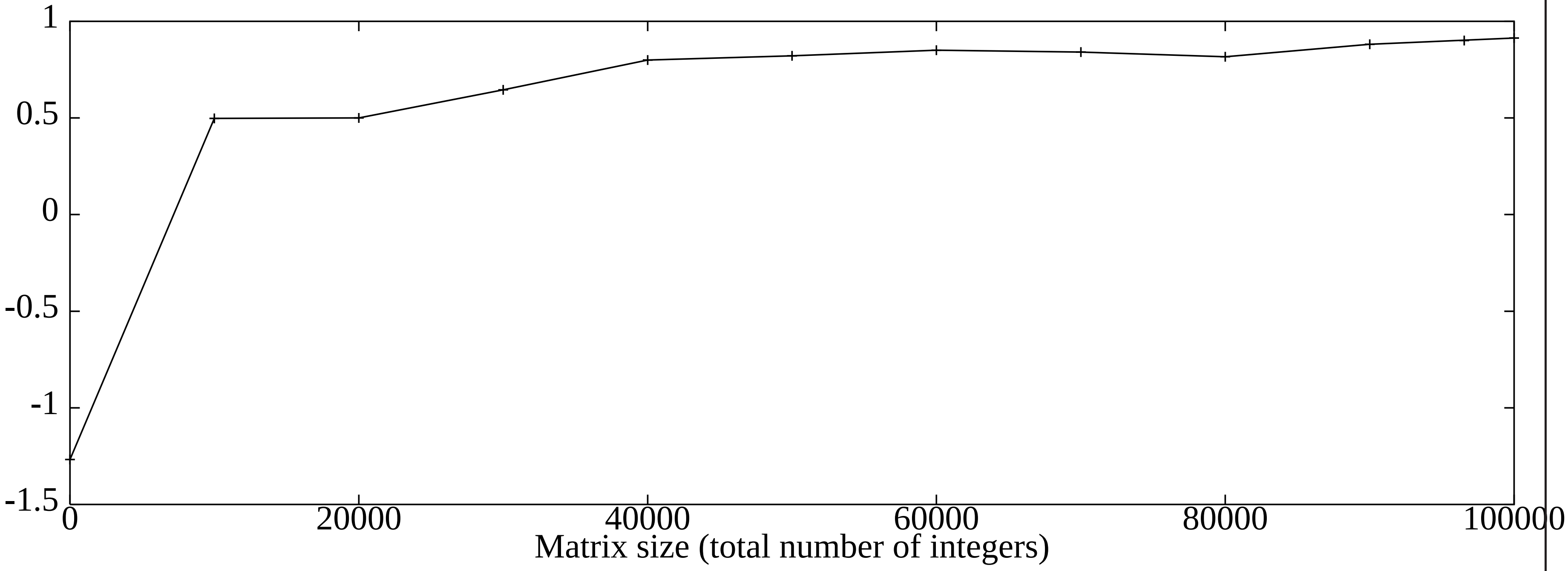
Benefit: Is the real gain close to optimal ?

$$Benefit\_Of\_Our\_Technique, B = \frac{Real\_Gain}{Optimal\_Gain} \qquad (1)$$

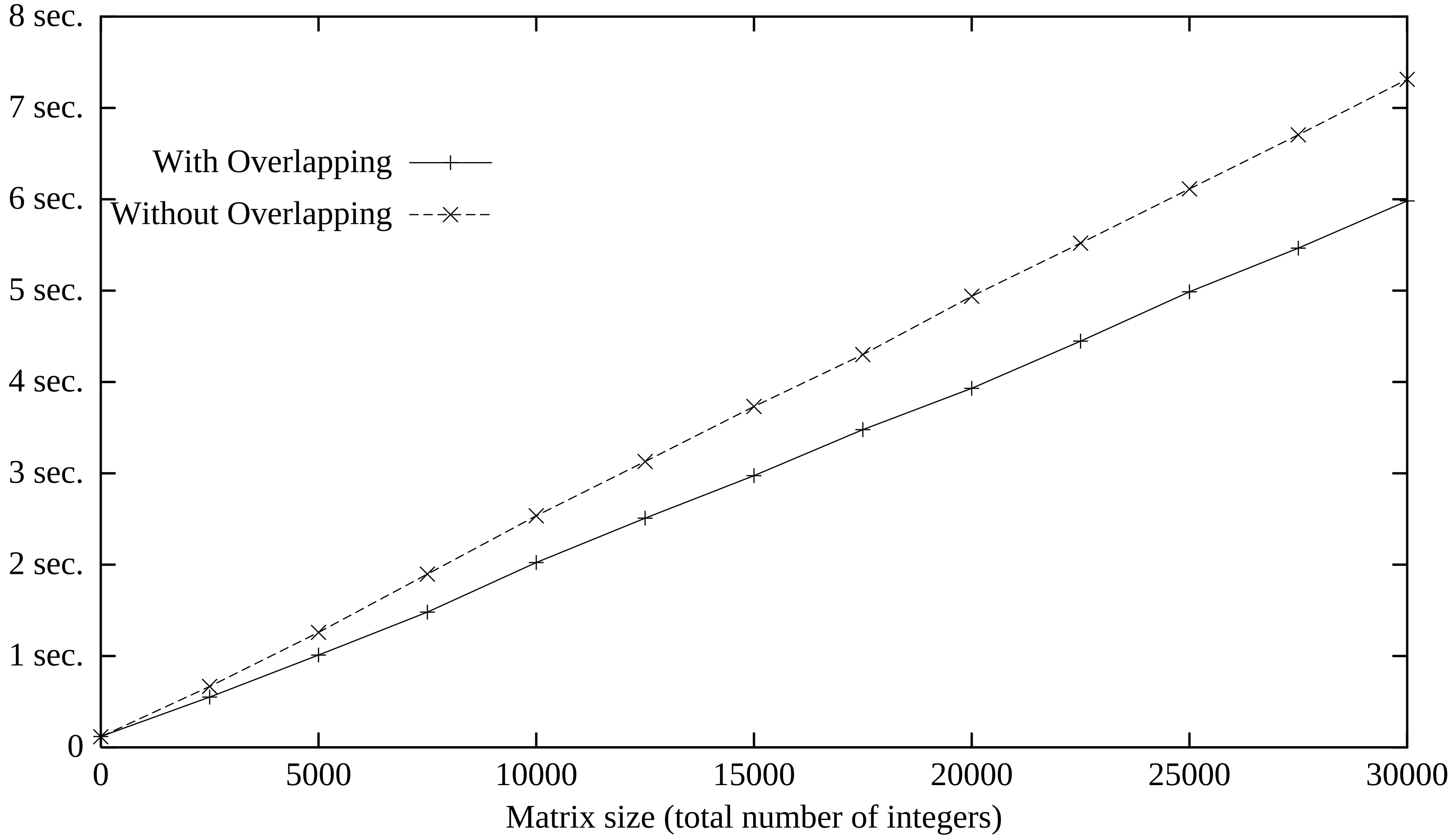
- in general $0 < B < 1$ ($B > 1$ is due to high network load variations)
- if $B < 0$, the overhead has not been masked. Why :
  - computation duration is too short w.r.t. transmission duration
  - or/and the duration to send data is too small (high-speed LAN, or/and too small amount of transmitted data).

$\Longrightarrow$ should not be the case for metacomputing applications !

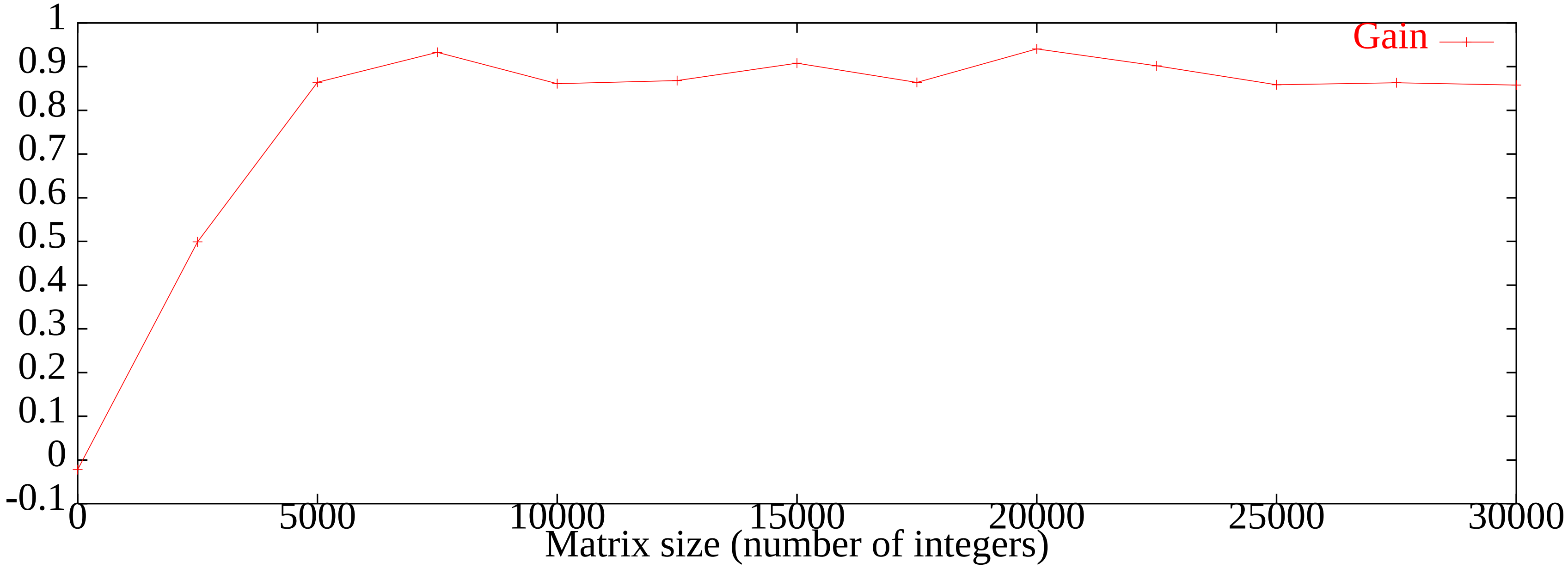
Benefit of the LAN-based experiment



One Experiment onto a WAN-based Grid



. . . and its positive and close to optimal gain



Conclusion

➤ Situations where the mechanism is relevant:

➢ Computation time of the already received parameters $\gtrsim$ transmission time of the subsequent ones.

➢ This transmission time must be high enough to neglect the network load variation and the small overhead of the mechanism.

➤ Almost no burden for the programmer . . .

$\Longrightarrow$ use the overlapping mechanism for every method call in metacomputing O.O applications !

For More Information ...

OASIS: Joint project CNRS/INRIA/Univ. of Nice-Sophia Antipolis

- Françoise Baude, Denis Caromel, Nathalie Furmento, David Sagnol

FirstName.LastName@sophia.inria.fr

- http://www.inria.fr/oasis/