A Framework for Experimenting with Structured Parallel Programming Environment Design

<u>M. Aldinucci</u>, S. Campa, P. Ciullo, M. Coppola,
M. Danelutto, P. Pesciullesi, R. Ravazzolo,
M. Torquati, M. Vanneschi, C. Zoccolo

Computer Science Dept. – University of Pisa – Italy

ISTI – National Research Council – Pisa - Italy

ParCo 2003 – Sept 4th, Dresden

Outline

- Motivations
- ASSIST Coordination Language
- ASSIST implementation (outline)
- Experimenting with ASSIST extensions
- Conclusions

Previous Experiences

Several environment for structured parallel programming:

- P3L (1991), C-based, fixed skeleton set: pipe, map …
- SkIE (1997), C/C++/F77/Java
- Lithium (2001), Java-based, macro data-flow, pipe, farm, map, D&C
- Many variants of them

- Lack of expressiveness
- Lack of flexibility
  - Any modification led to extensive changes within compiler & run-time support

- ASSIST: A Software development System based on Integrated Skeleton Technology
- Aiming at
  - Providing flexible structured parallel programming environment
  - Achieving efficiency and portability
  - Targeting clusters (homogeneous and eterogeneous)
  - Being usable to perform experiments in structured parallel SW development systems design

ASSIST Approach

Evolution of the Structured Parallel Programming Approach

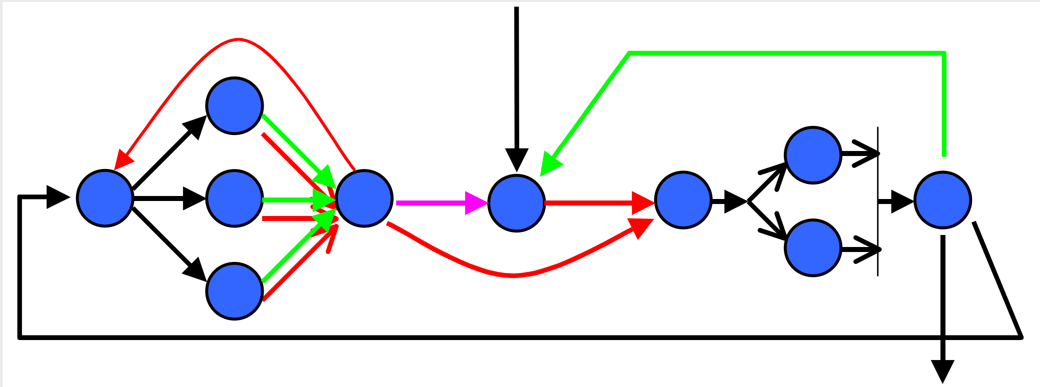
Parallel Coordination Language

- classical skeletons and
- new composition forms
- coordinate sequential code modules

w.r.t. previous work, enhanced support for

- irregular and data-intensive applications
- complex, variable interaction patterns

ASSIST Fundamentals (1)



Sequential modules

- written in several host languages (C, C++, Fortran, Java)

Arbitrary Composition

generic graph

- stream-oriented
- both data-flow and nondeterministic with state

Not only fixed-pattern Parallel Skeletons ...

- classic task- and data-parallelism forms: pipeline, farm, loop

ASSIST Fundamentals (2)

Programmable Skeleton *parallel module*

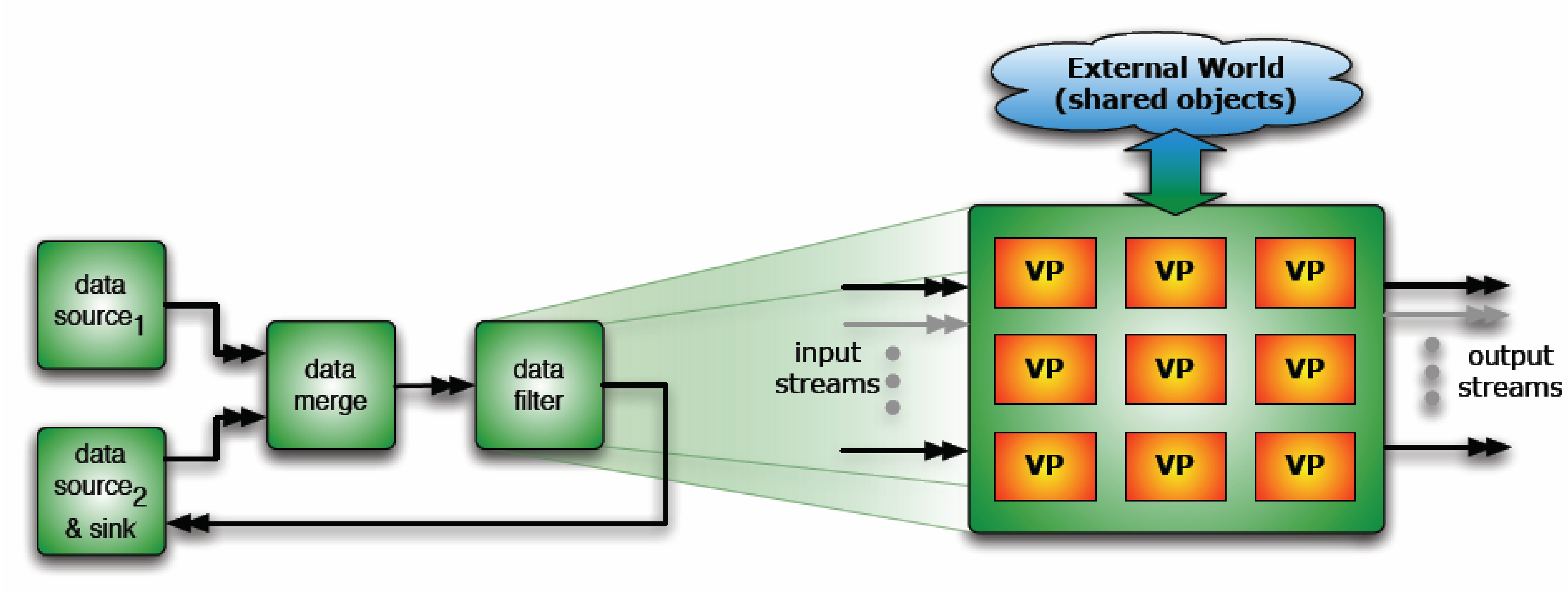
- both task and data parallel
- supports (local/global) module state
- variable comunication patterns
- nondeterminism, concurrency

Heterogeneous Resources *external objects*

- externally managed, standard protocols
- export/import SW components

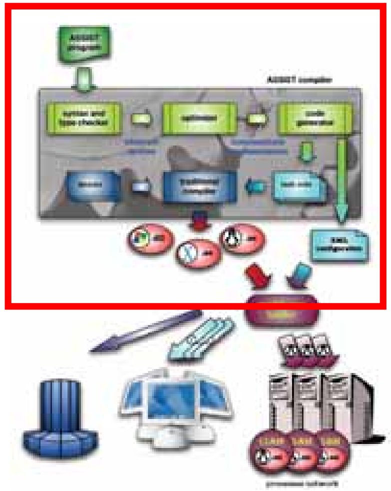
Modules, streams, non-determinism, inter/intra-parallelism, shared objects

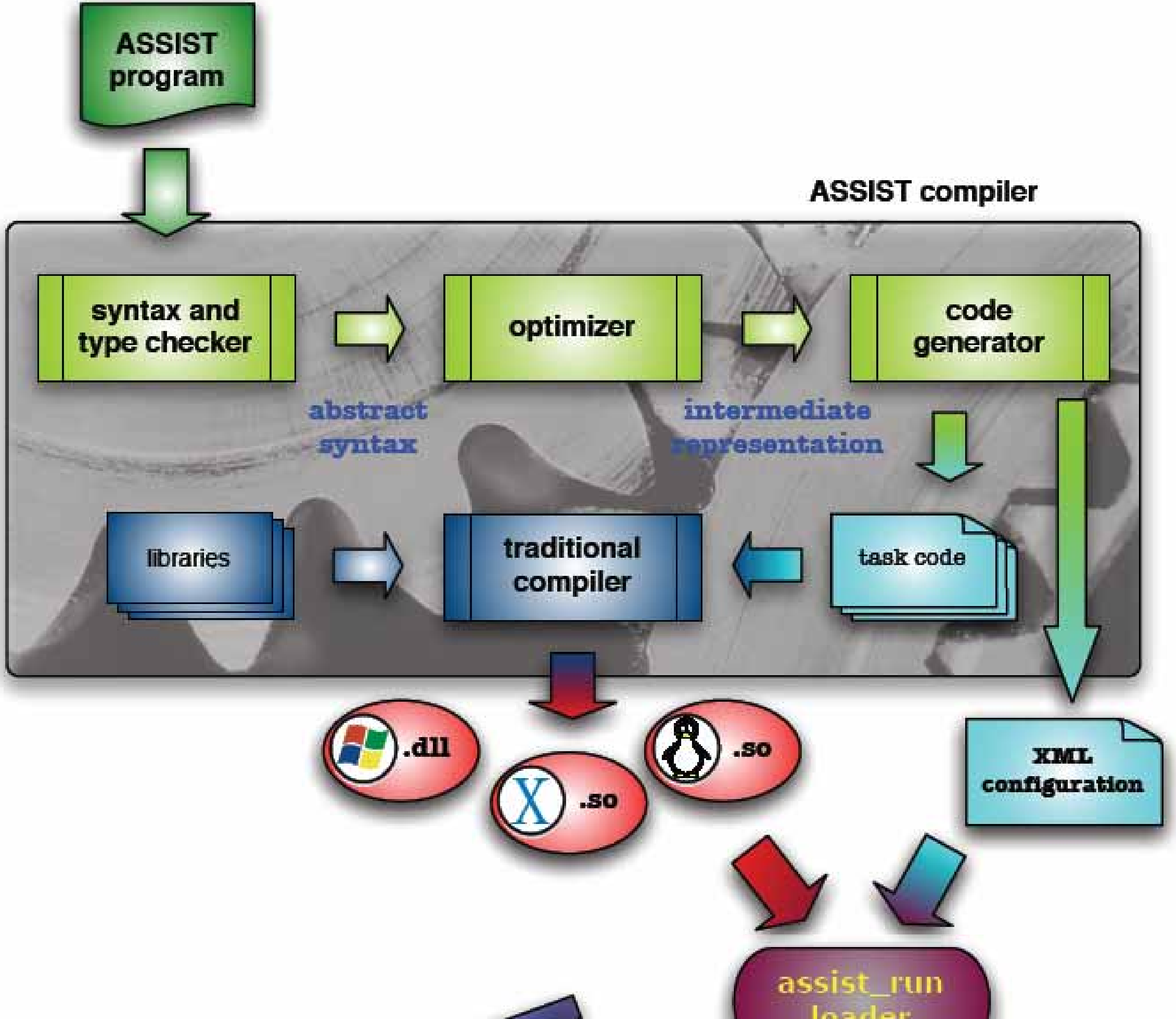
DI1



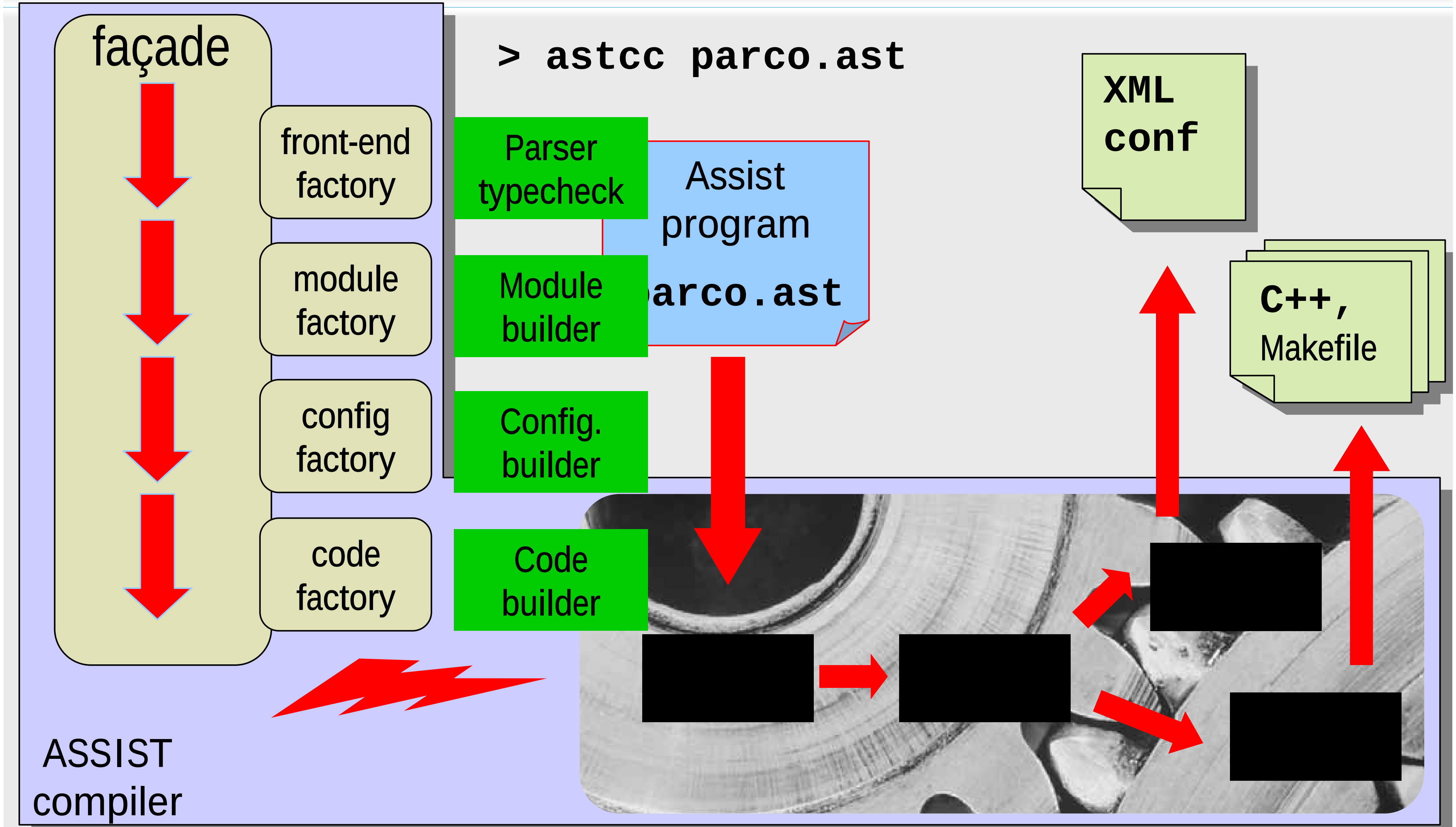
DI1 programmer may structure parallel application as a generic graphs of modules.
Modules are connected by means of data streams.
Non-deterministic control is provided to accept inputs from different streams and explicit commands are provided to output items on the output streams.
Parallelism both among modules and within modules
DipInf; 03/09/2003

ASSIST – the big picture

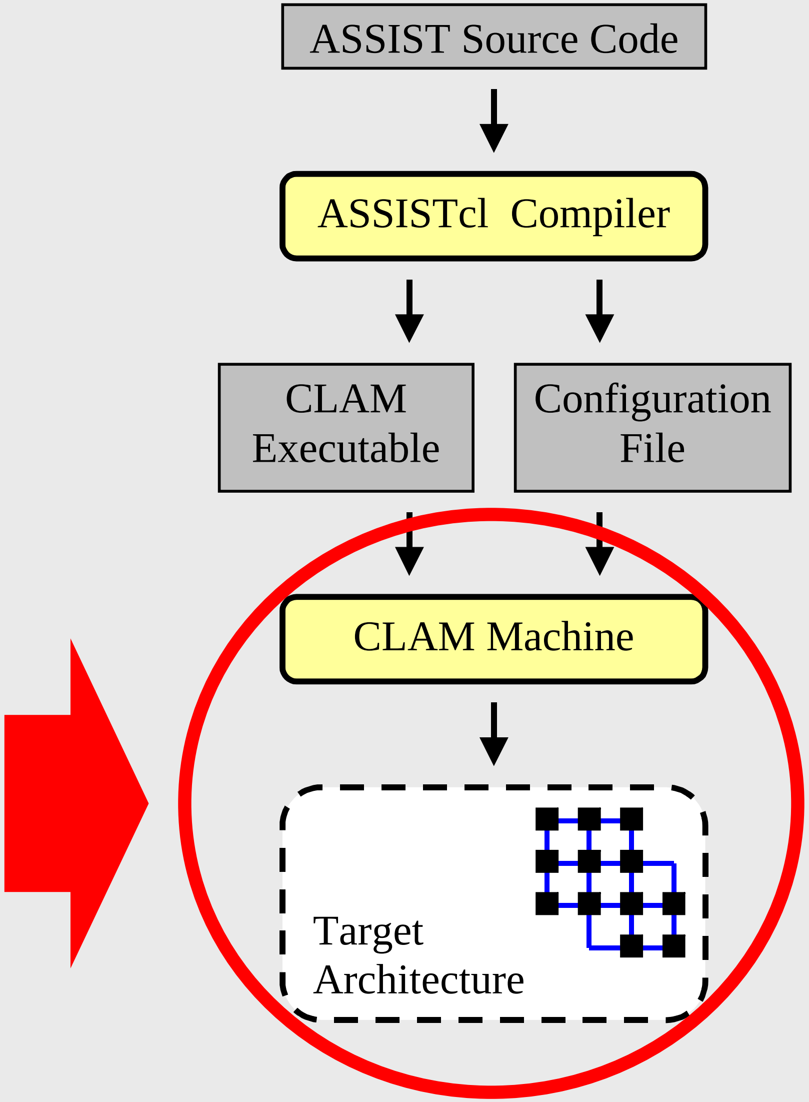


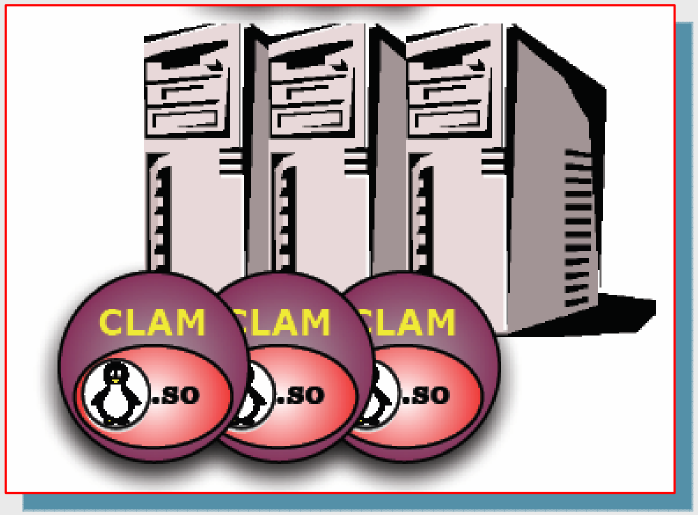


Design patterns based

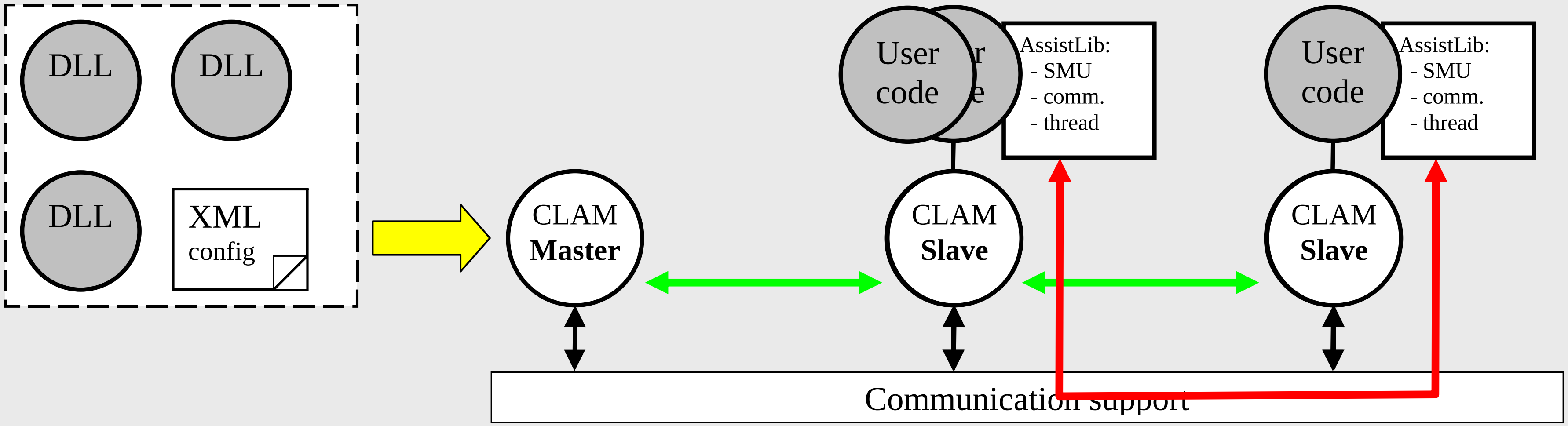


CLAM


ASSIST Source Code
ASSISTcl Compiler
CLAM
Executable
Configuration
File
CLAM Machine
Target
Architecture


CLAM
.so
CLAM
.so
CLAM
.so

XML Configuration and Loading

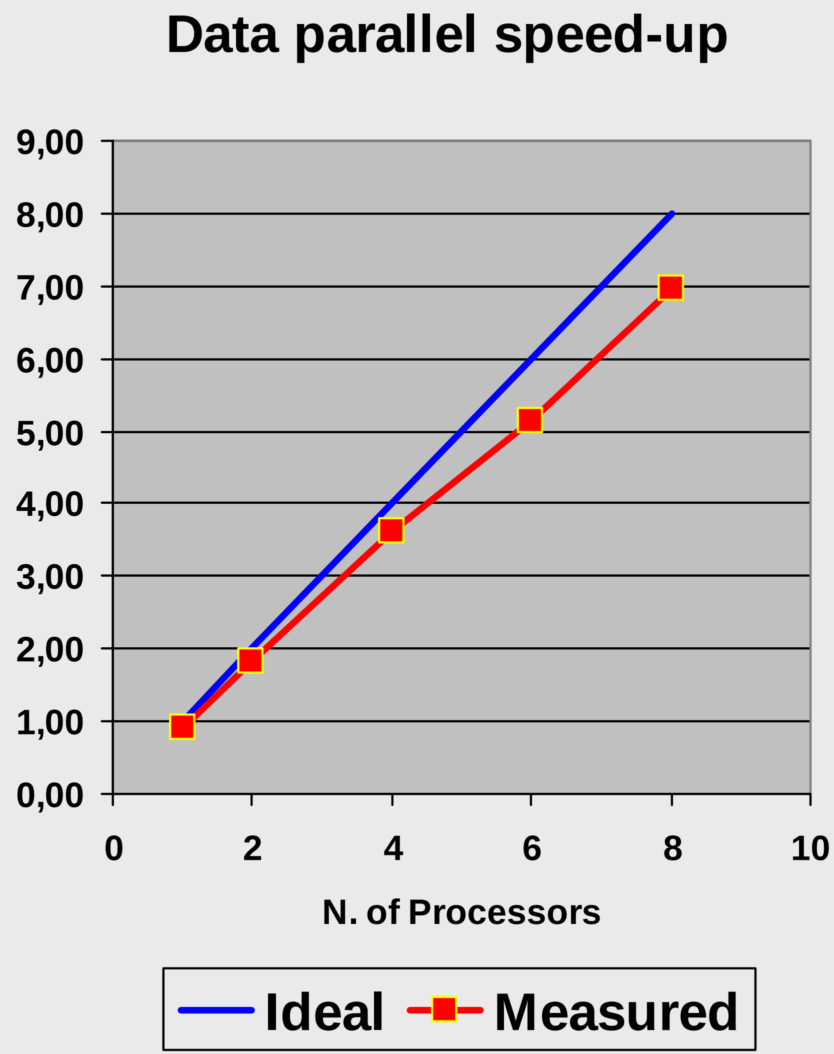


- **`ast_run`**
- A master CLAM is executed
- Several CLAM slaves are executed
- CLAMs maps “processes” to computing resource

Performance Benchmarks

Data-Parallel Benchmark (Shortest Path)

- 2-D matrix 400x400
- partitioned row-wise
- variable communication stencil
- 8 x Pentium 4, Gbit Eth

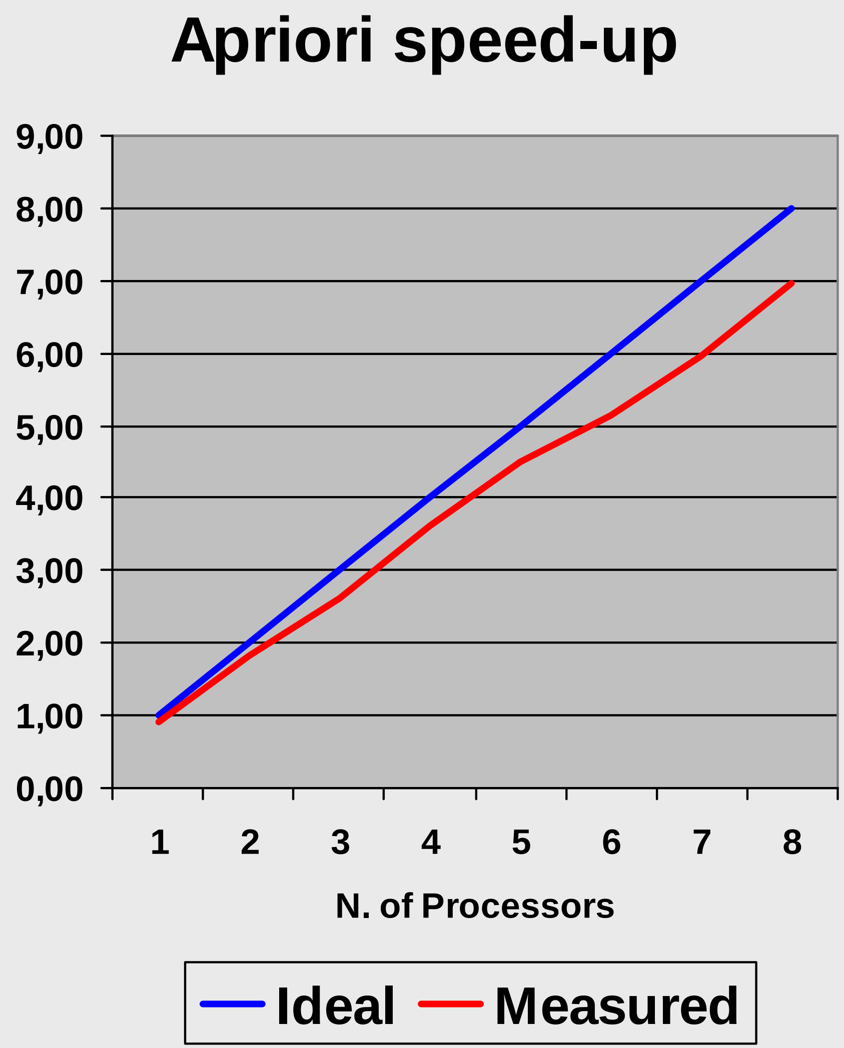


Performance Benchmarks

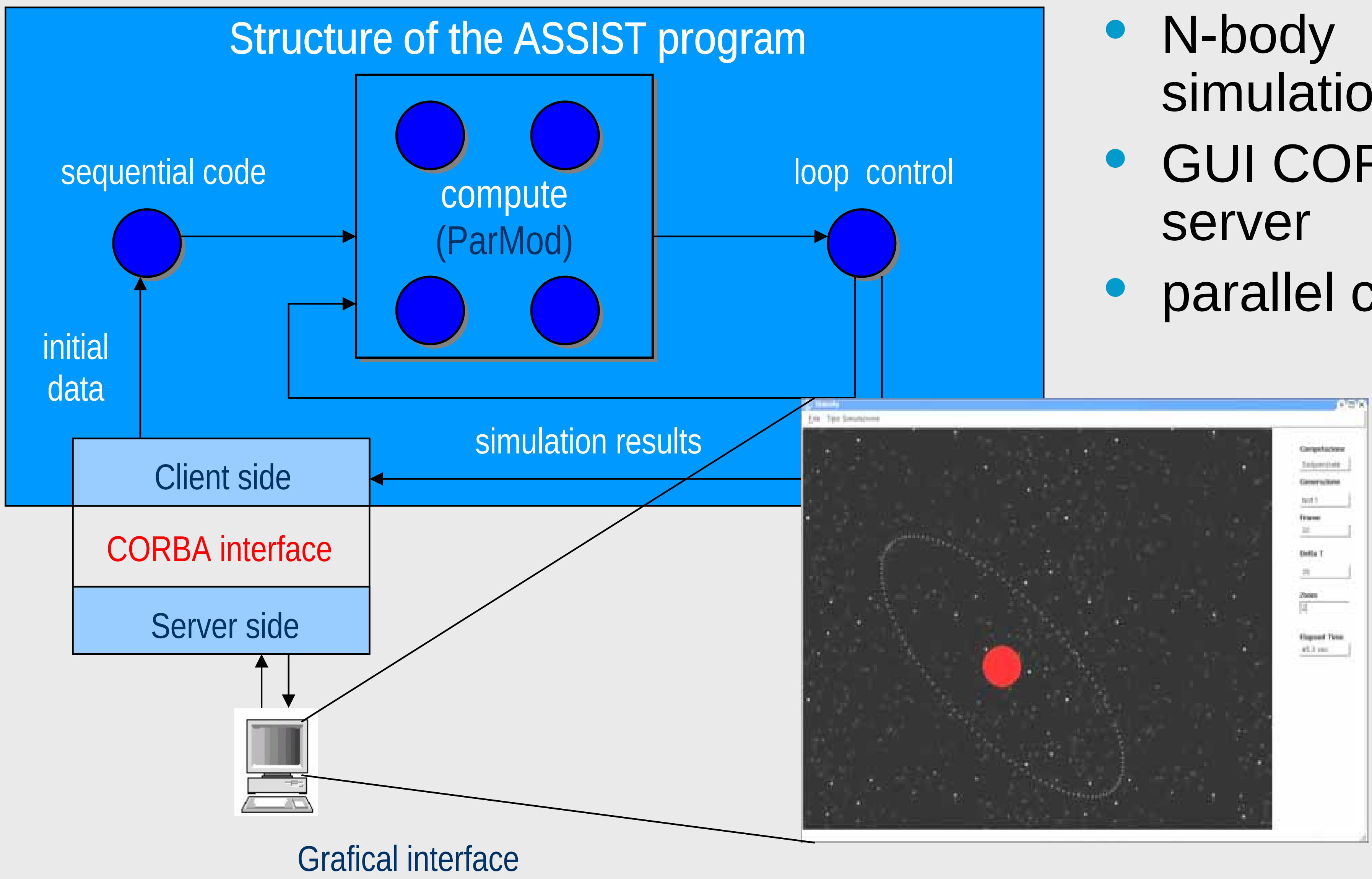
Parallel Partitioned Apriori

- Mainly stream-parallel
- Computation intensive, well balanced
- dataset > 160 Mb
- regular I/O pattern

- 8 x Pentium 4, Gbit Eth



Integration with CORBA Code

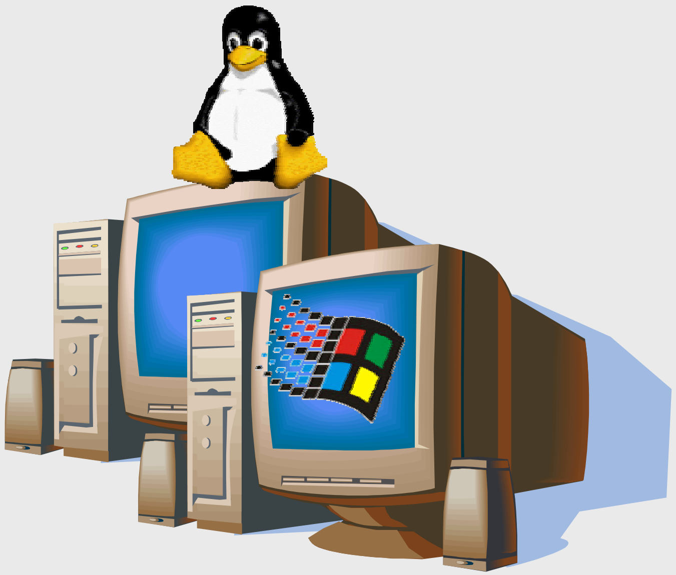


- N-body simulation
- GUI CORBA server
- parallel client

Experimenting with extensions

1. Targeting heterogeneous COWs
2. Integrating parallel MPI libraries
3. Targeting the GRID (ongoing)

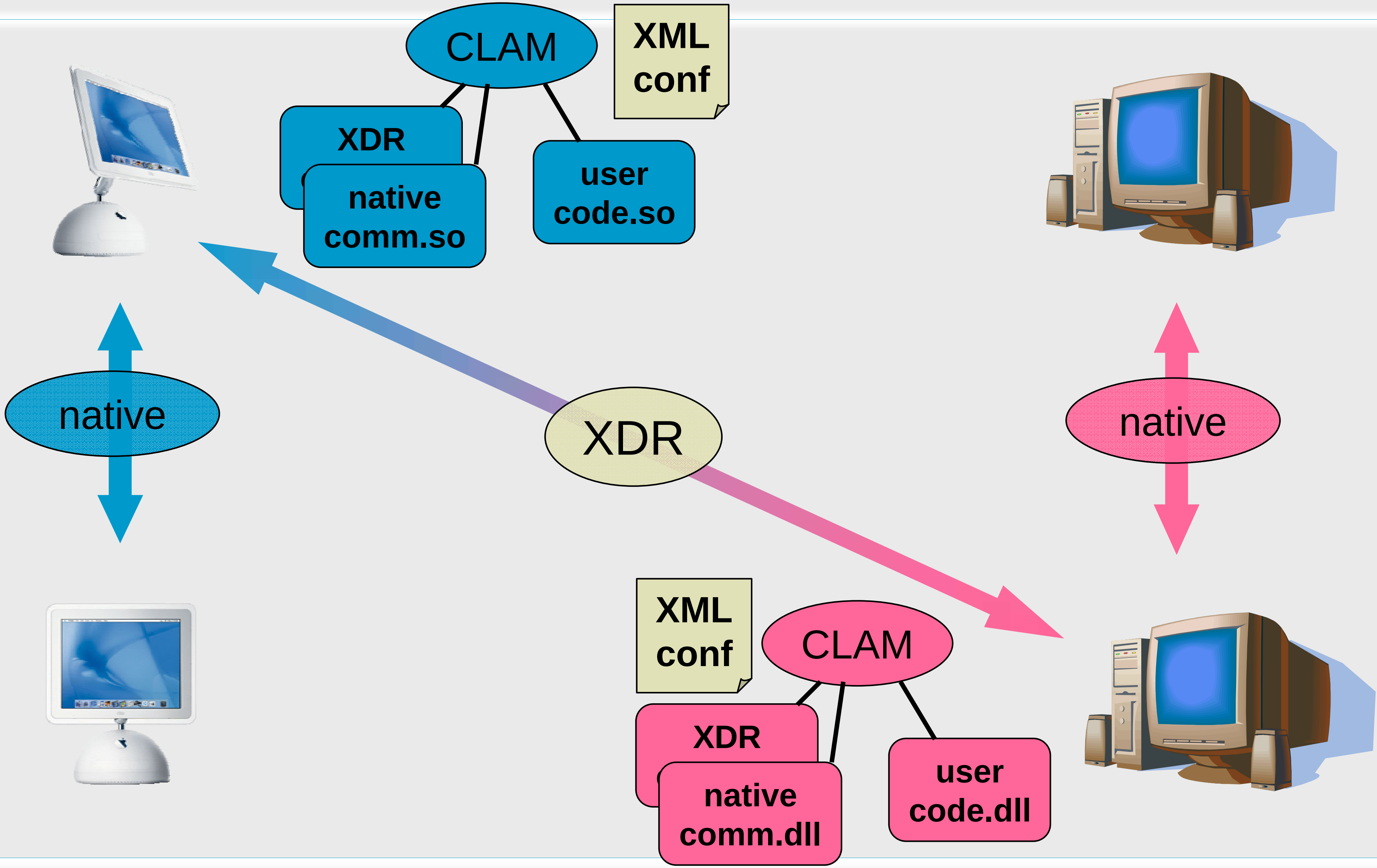
Targeting heterogeneous COWs



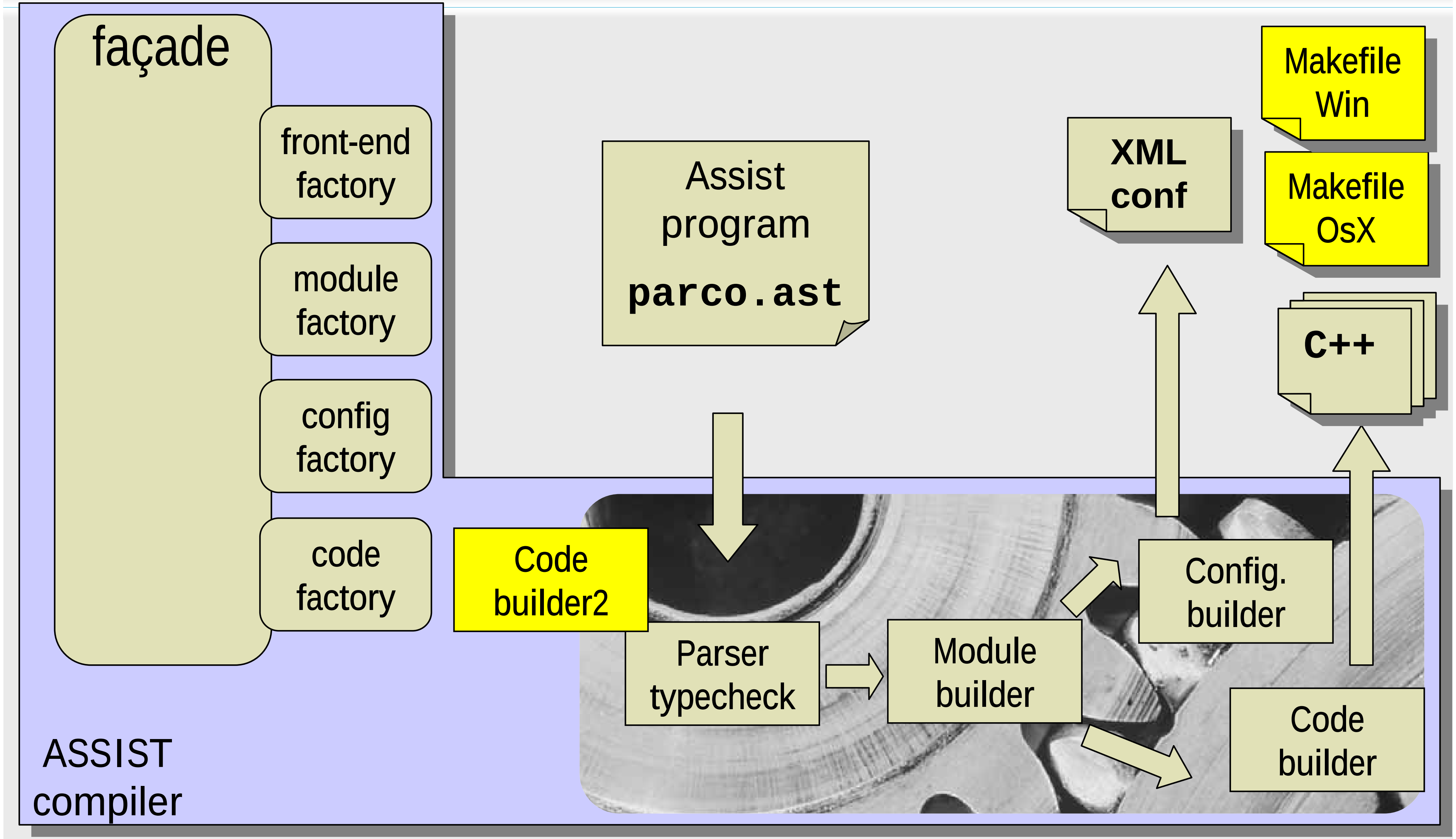
XDR + dynamic loading

- Initially targeted to homogenous COWs
- Different versions of comm code
  - raw and XDR communications
  - compiled for different architectures
  - as .dll & .so objects
- Make decisions dynamically
  - CLAM + XML match the correct lib w.r.t. communication ends
  - Use the fastest lib

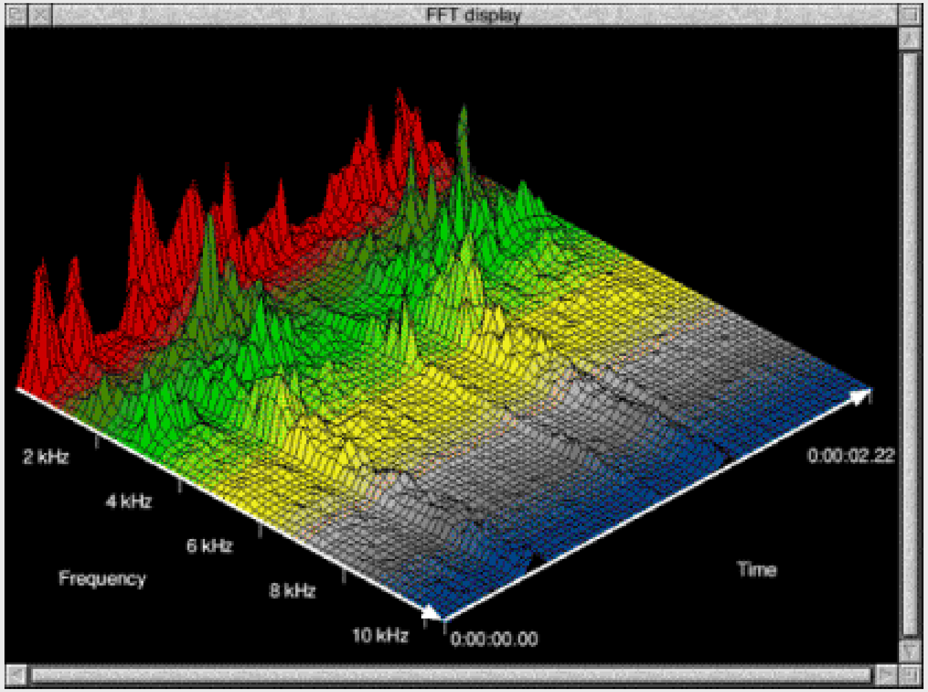
Choose the fastest method



Just enrich the code factory



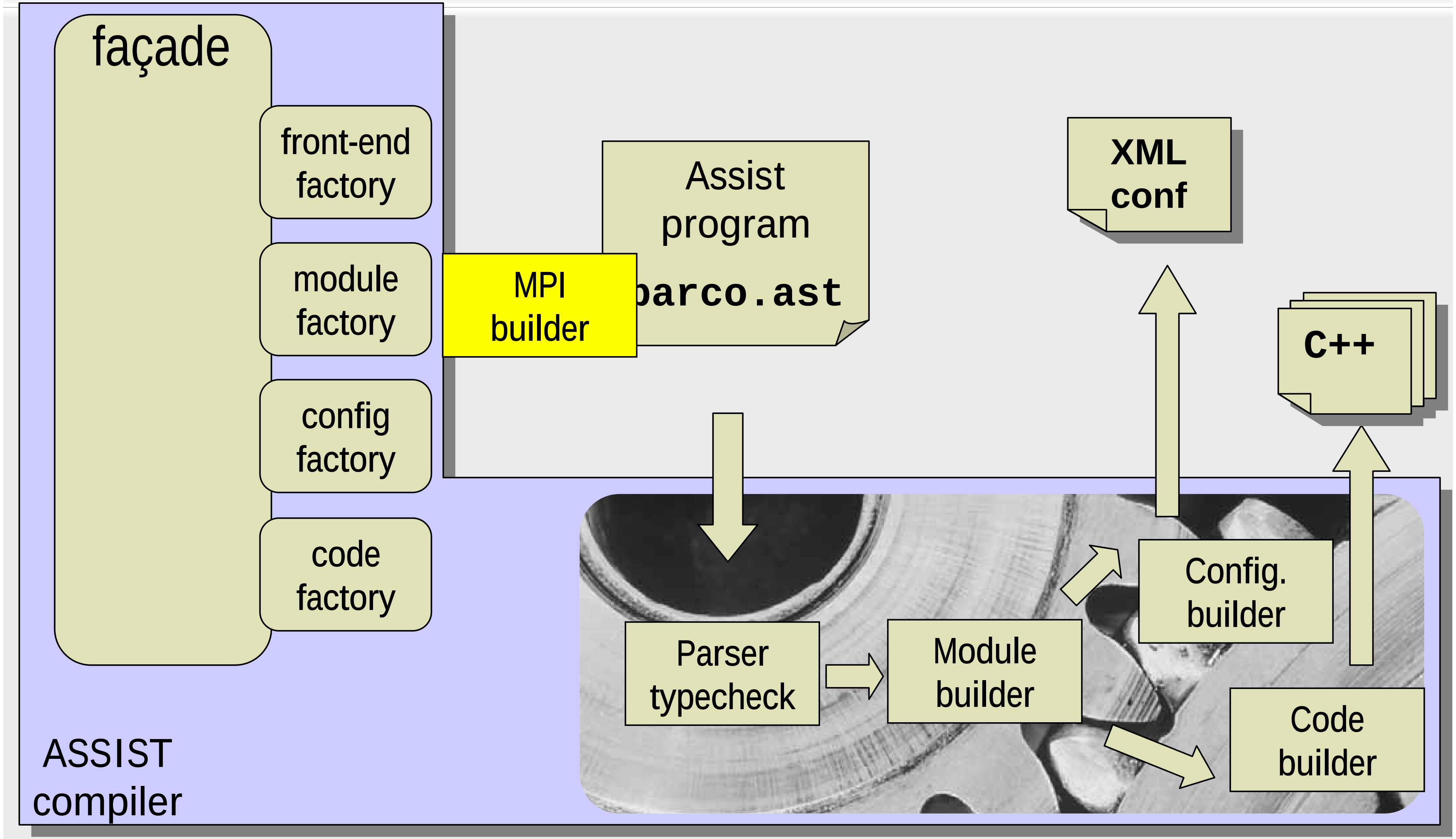
Add parallel MPI libraries



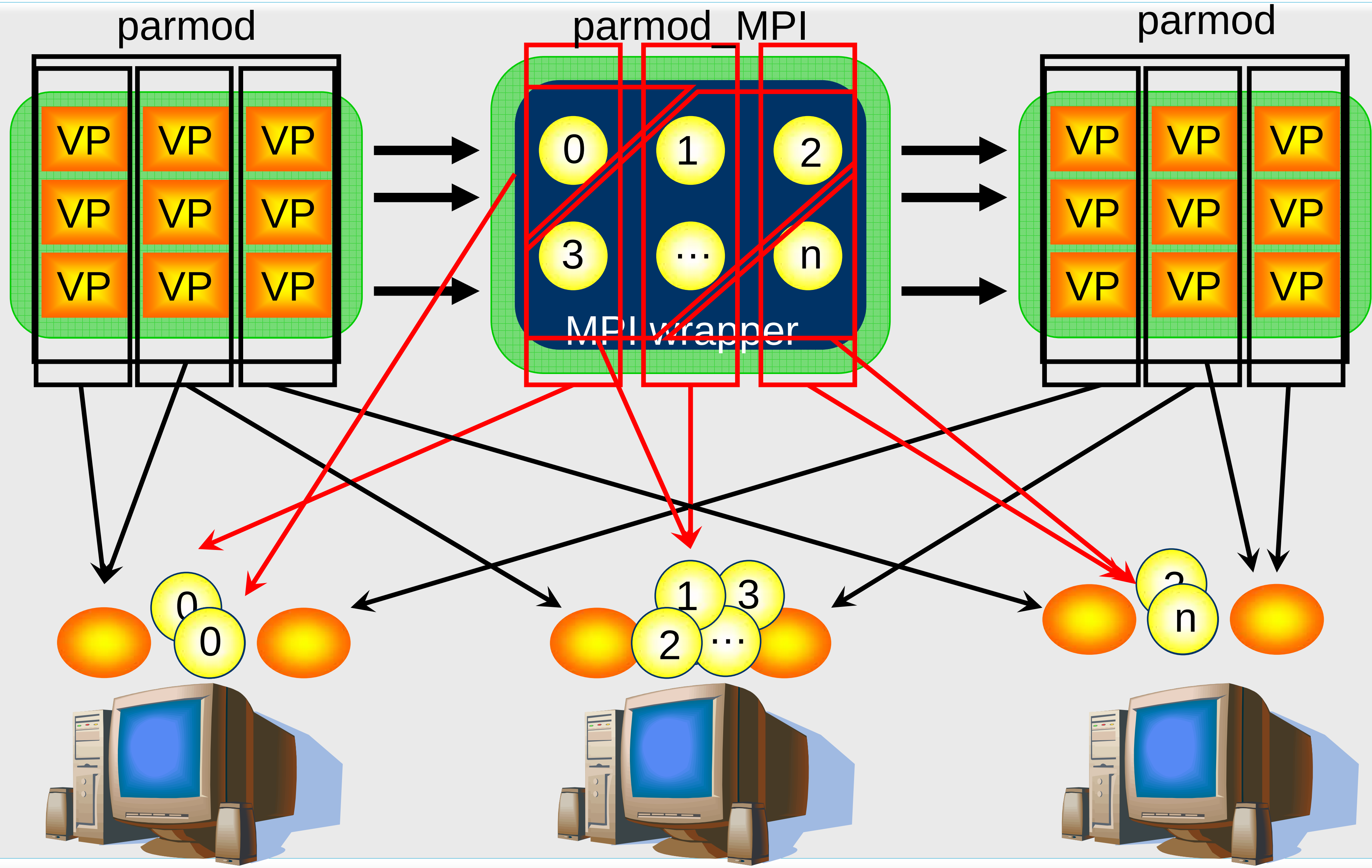
Add parallel MPI libraries

1. Define a new parmod flavor
   - Acting as MPI program container
2. Write a MPI wrapper program
   - exchanging in/out with parmod interfaces
   - calling the library
3. Modify **`mpirun`** to interact with CLAM
   - get from CLAM mapping information
4. Extend “module factory”
5. ScaLAPACK, PAMIHR [PC28(12):2002]

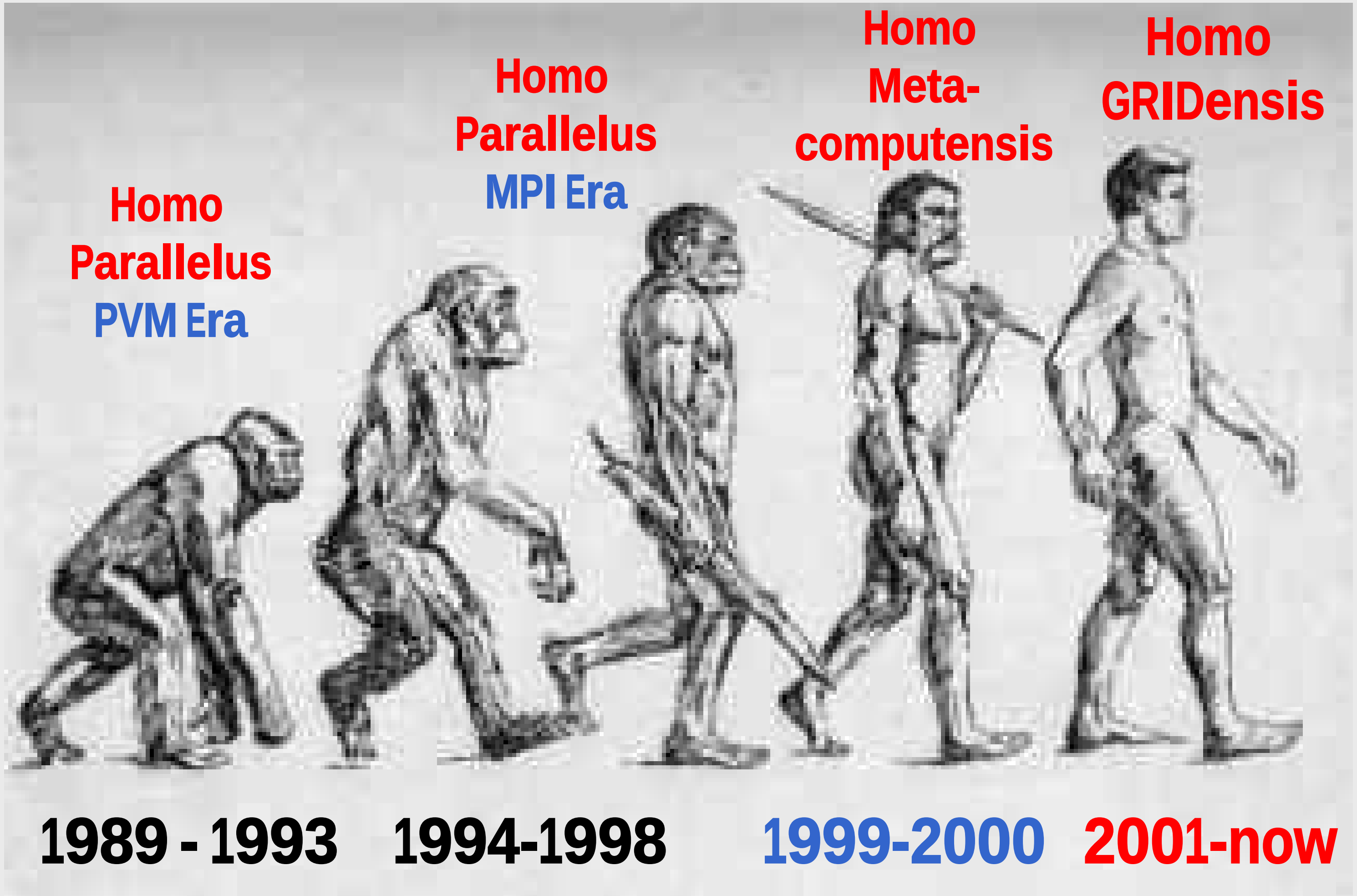
Just enrich the module factory



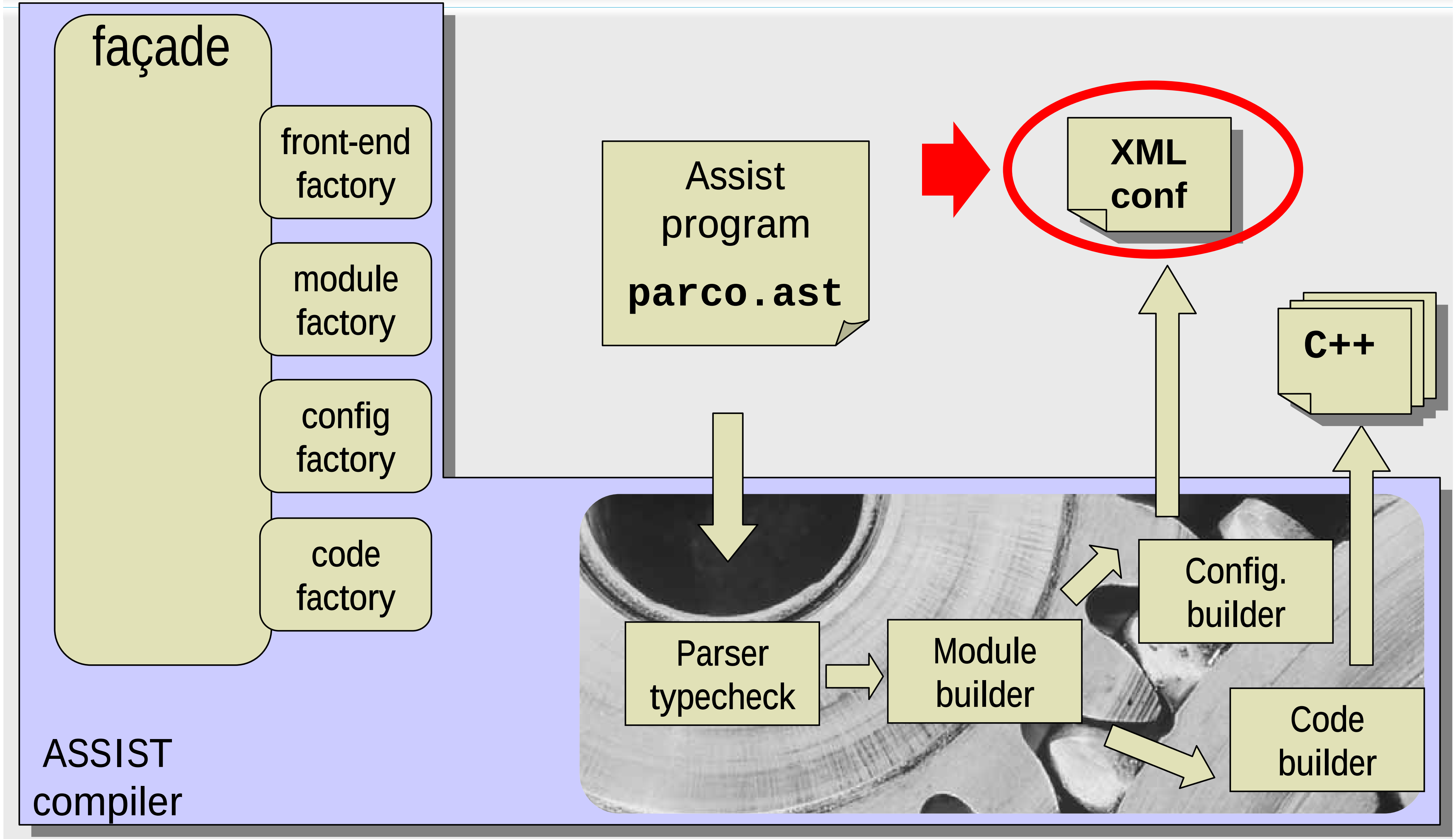
MPI integration summary



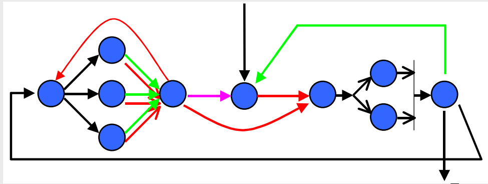
Targeting the GRID



Targeting the GRID



XML conf

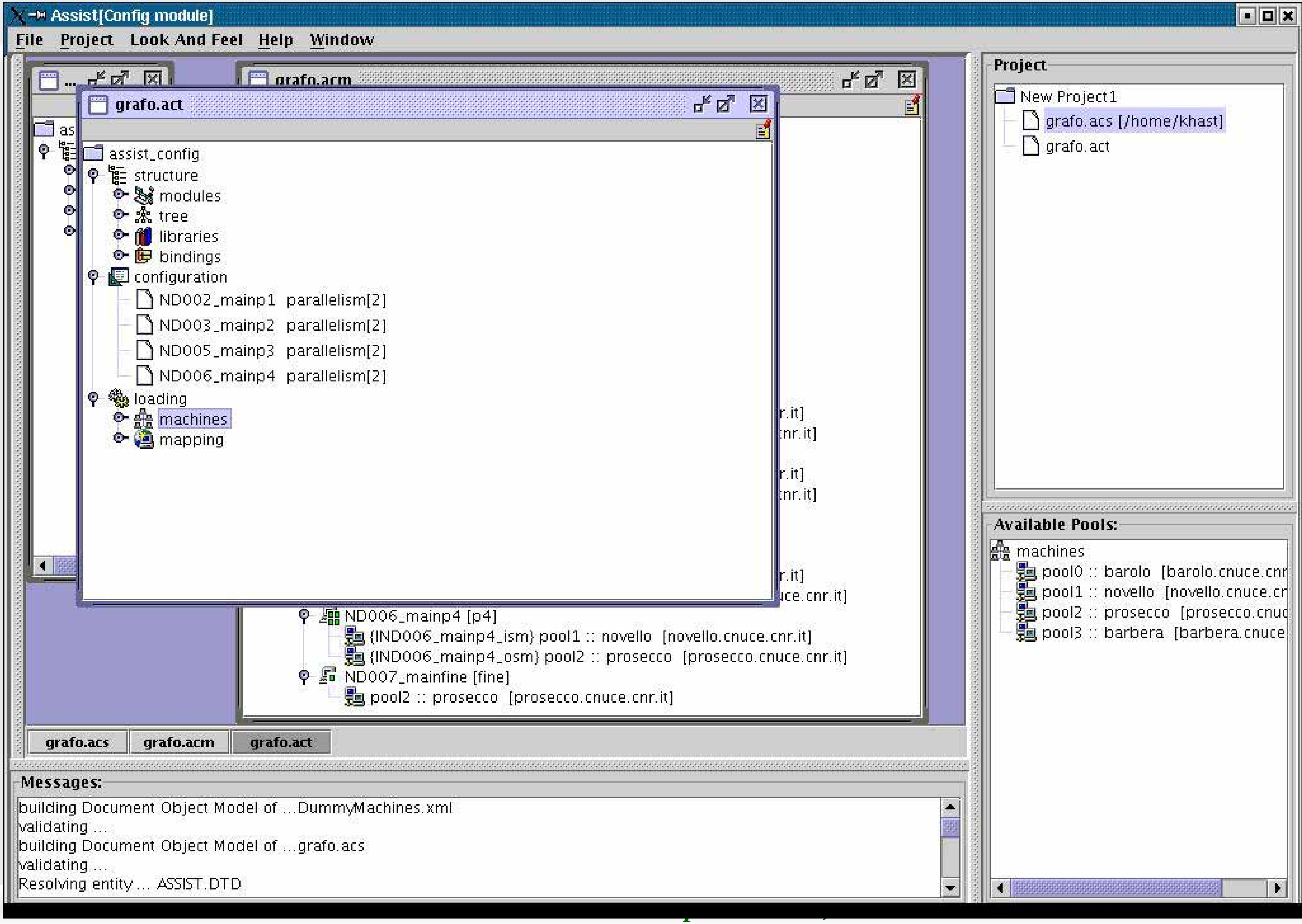
- modules list (parallel activities)
- modules graph 
- pathnames, lib-names, code-names
- lib-modules bindings

static

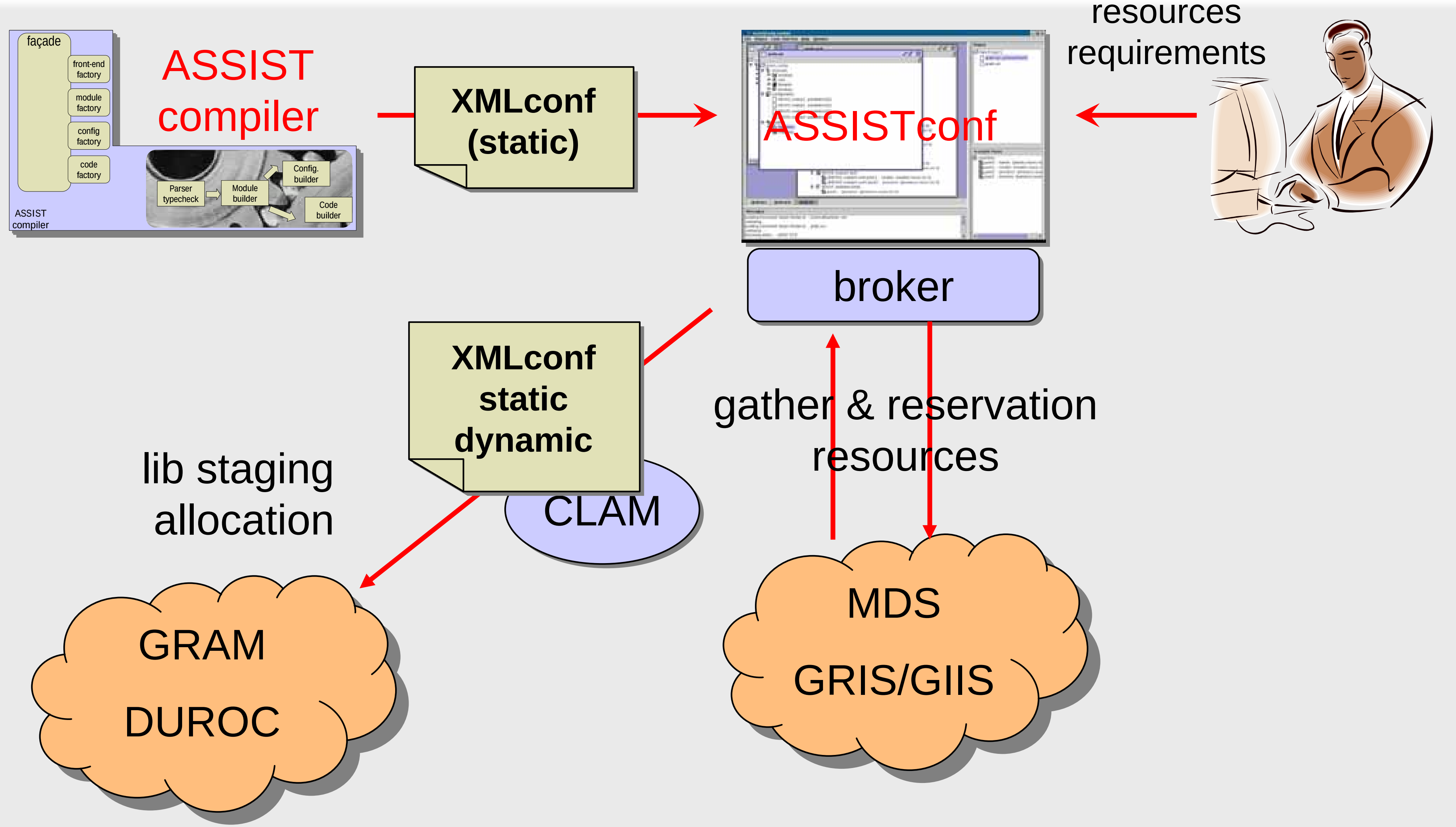
- machine names
- modules parallel degrees
- modules-machines mapping

dynamic

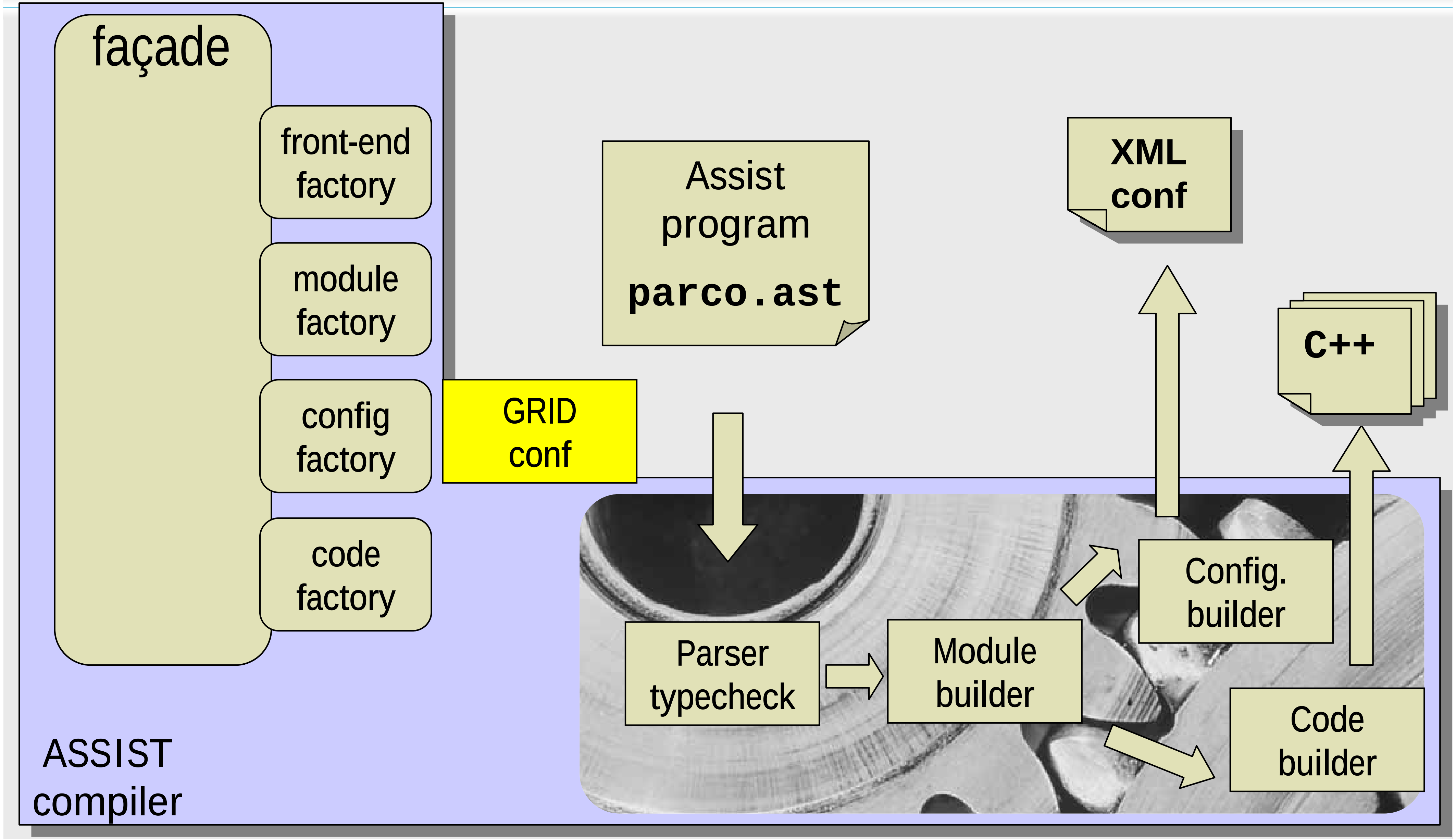
ASSISTconf


Assist[Config module]
File Project Look And Feel Help Window
grafo.acm
grafo.act
assist_config
structure
modules
tree
libraries
bindings
configuration
ND002_mainp1 parallelism[2]
ND003_mainp2 parallelism[2]
ND005_mainp3 parallelism[2]
ND006_mainp4 parallelism[2]
loading
machines
mapping
ND006_mainp4 [p4]
{IND006_mainp4_ism} pool1 :: novello [novello.cnuce.cnr.it]
{IND006_mainp4_osm} pool2 :: prosecco [prosecco.cnuce.cnr.it]
ND007_mainfine [fine]
pool2 :: prosecco [prosecco.cnuce.cnr.it]
grafo.acs
grafo.acm
grafo.act
Messages:
building Document Object Model of ...DummyMachines.xml
validating ...
building Document Object Model of ...grafo.acs
validating ...
Resolving entity ... ASSIST.DTD
Project
New Project1
grafo.acs [/home/khast]
grafo.act
Available Pools:
machines

ASSIST-G


façade
front-end factory
module factory
config factory
code factory
ASSIST compiler
Parser typecheck
Module builder
Config. builder
Code builder
ASSIST compiler
XMLconf (static)
ASSISTconf
resources requirements
broker
XMLconf static dynamic
CLAM
gather & reservation resources
lib staging allocation
GRAM
DUROC
MDS
GRIS/GIIS

Just enrich the config factory



Summary

- Tested over real-world applications
  - Data-mining (C4.5, apriori, …), computational chemistry & numerical kernels, digital grading, MPEG encoders…
- Support interoperability
  - May act as CORBA client/server, MPI, PVFS, several DSM
- High-performance
  - Very good speedup in many cases
- Easily extendable
  - Design pattern based
  - Robust

Ongoing work

- Full GRID support
  - First prototype based on globlus 2 [euromicro03]
  - Within *Grid.it, CoreGRID, …*
- Enhanced support for highly-irregular apps & dynamic data structures [PPL(to appear)]
- Standardization of components
  - Already based on component technology
  - Match high-performance with standards

THANK YOU!

Questions?