

# **MPI Support on Opportunistic Grids based on the InteGrade Middleware**

Marcelo de Castro Cardozo  
Fábio M. Costa

**LAGRID'08: Latin American Grid  
2 International Workshop**



# Summary

Overview

MPI and MPICH2

InteGrade

MPICH-IG

Evaluation Results

Related Work

Final Remarks



# Intro

Grids for HPC: widely available resources

Scavenging grids: opportunity to use idle resources for parallel apps

Extra complexity

- dynamic environment: resources come and go
- resource location and scheduling
- reliability and FT, security, heterogeneity

Hide complexity from parallel app programmer

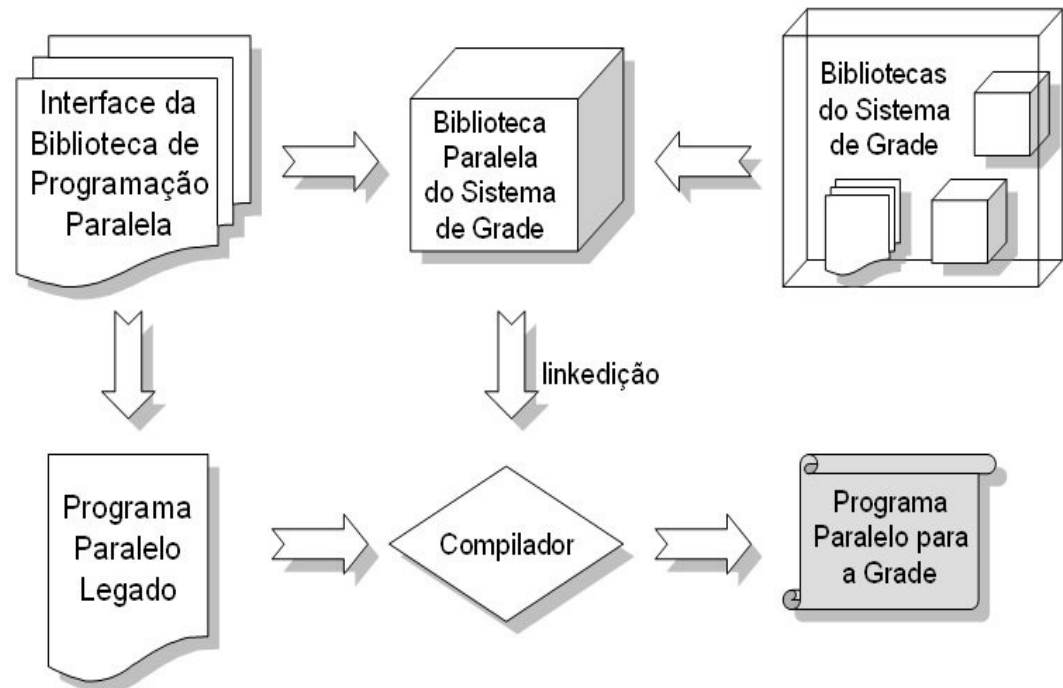


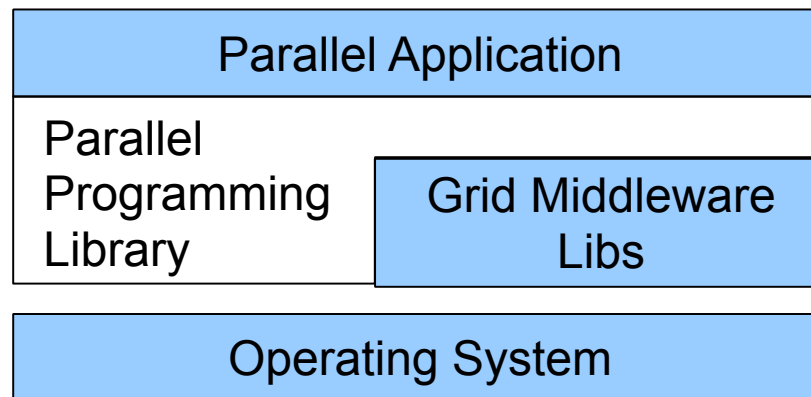
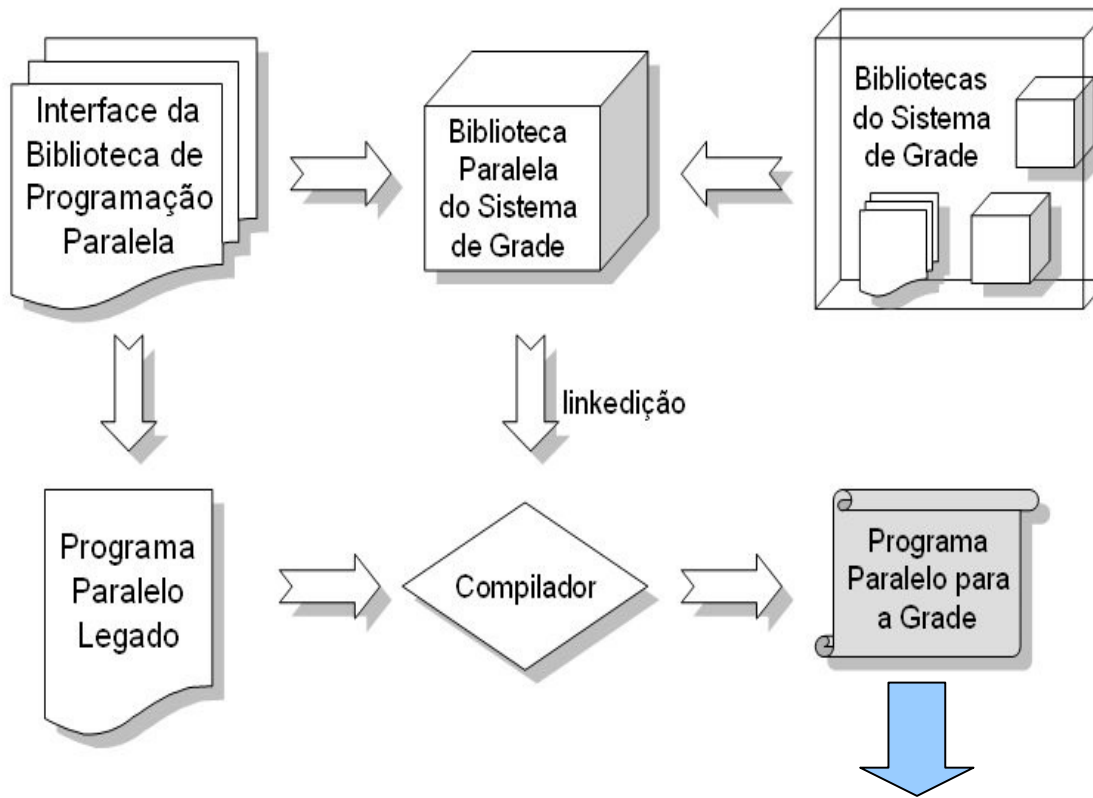
# Overall Approach

Modify parallel programming library

- use grid middleware services instead of native services

Avoid changes  
to legacy apps  
and to the  
grid middleware





} Parallel Program enabled to Run on the Grid



# MPI

De facto standard for parallel applications programming (MPI2)

Library functions for

- point-to-point communication with several different semantics
- group communication and group management
- management of execution state
- support for DSM
- dynamic spawning of processes



# MPICH2

Popular implementation of MPI2

Layered architecture to ease portability

- only the lower layers are platform-dependent
  - IPC and process management

ADI – Abstract Device Interface

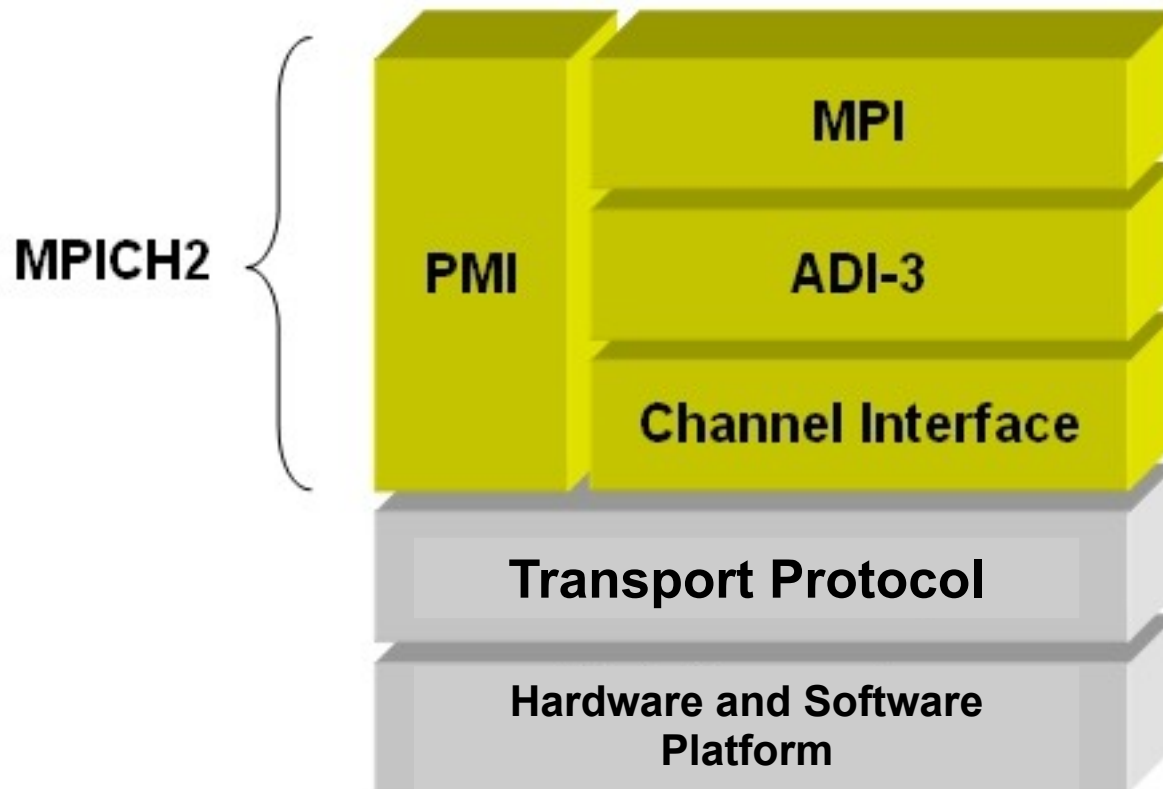
- Expose high-level functions

Channel Interface: low-level functions

PMI – Process Management Interface

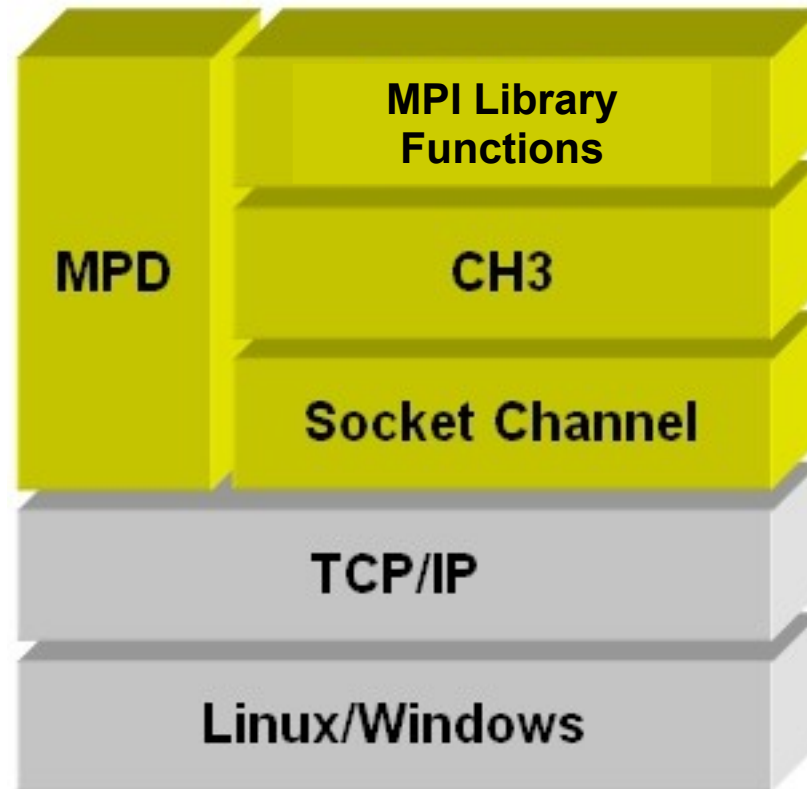


# MPICH2 Layered Architecture





# MPICH2 Implementation




# Overview of InteGrade

Opportunistic Grids

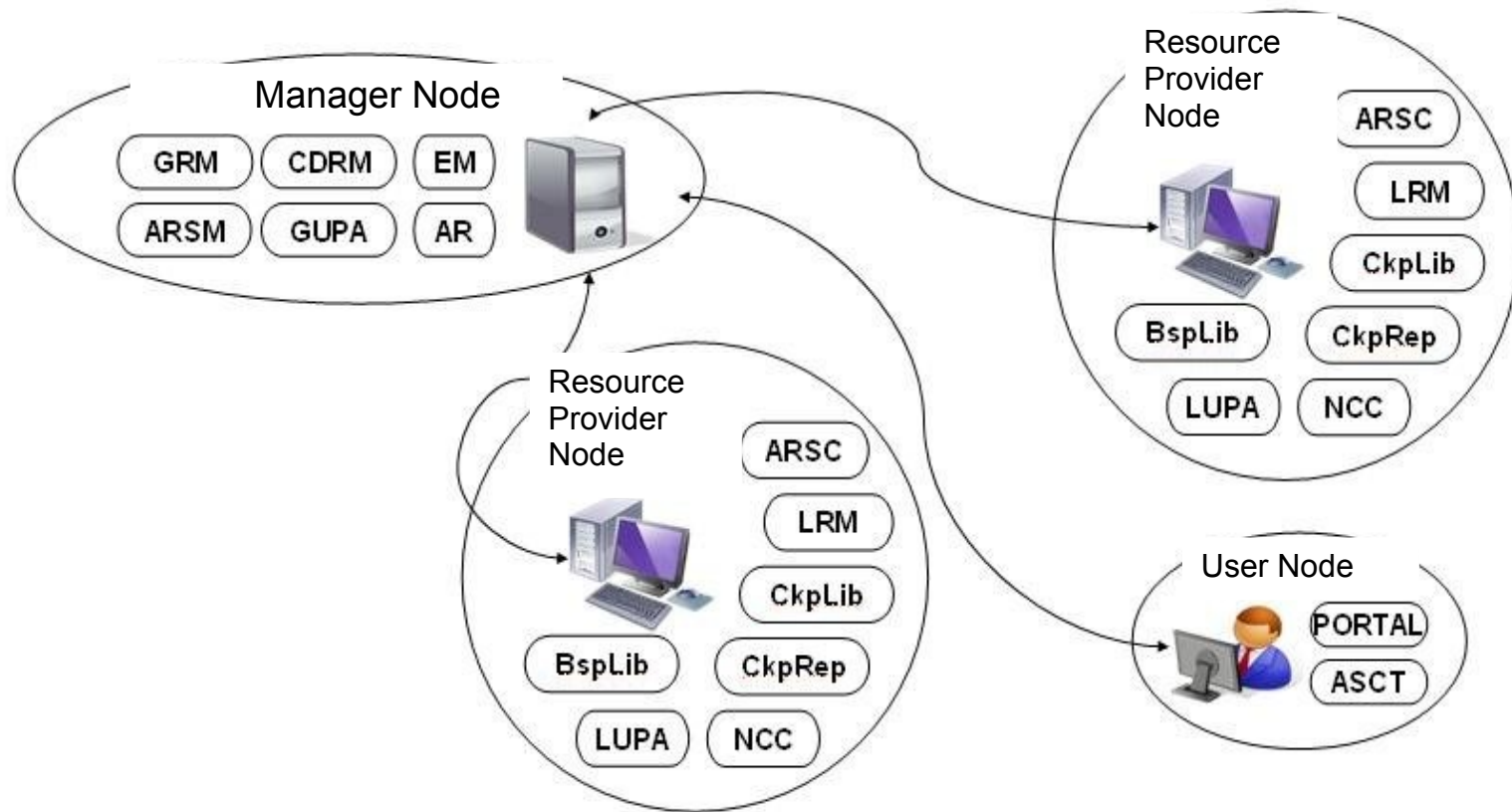
Object-oriented implementation

Support for different application programming models:

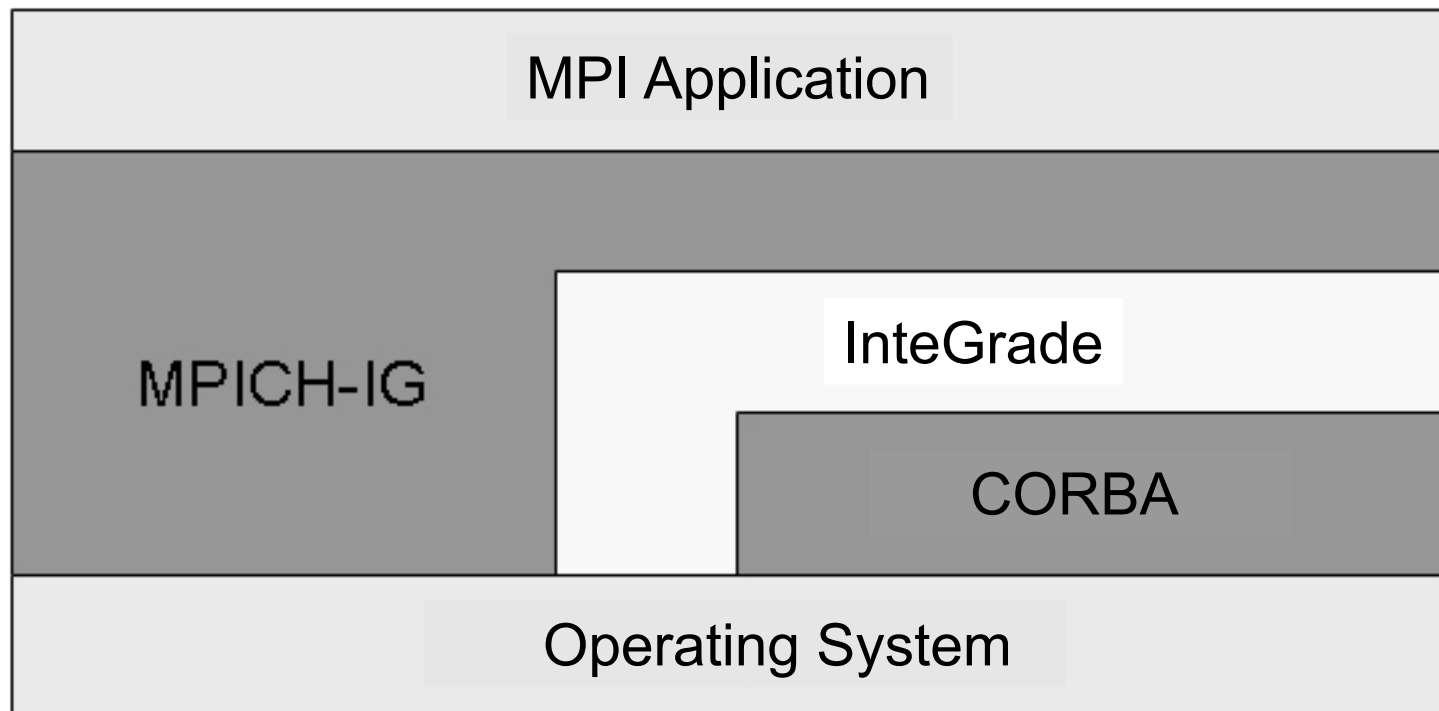
- Sequential
- Bag-of-Tasks
- BSP
- MPI 



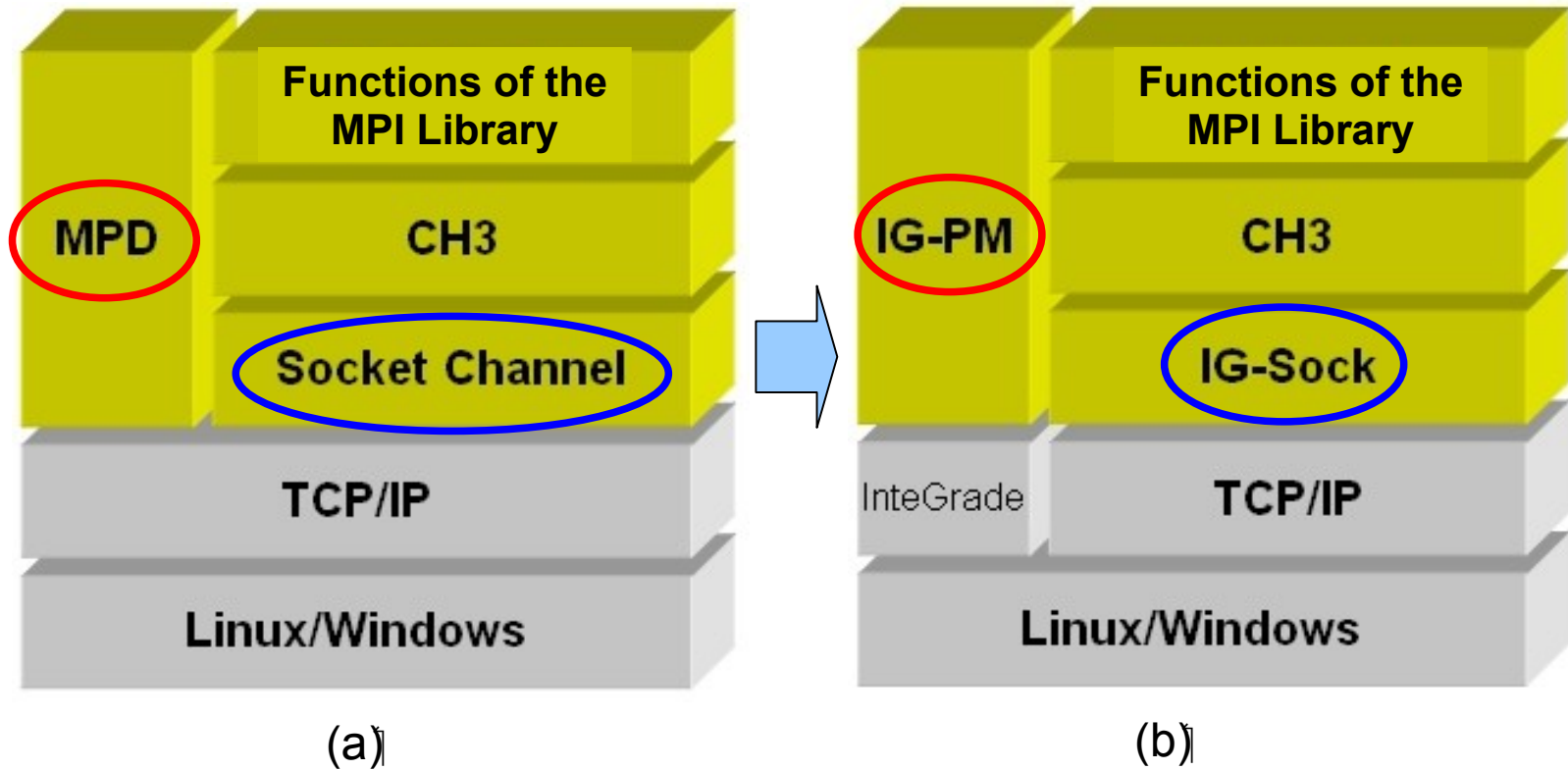
# InteGrade Components



# MPICH-IG Overall Implementation



# MPICH2 and MPICH-IG



# MPICH-IG Main Components

## IG-PM

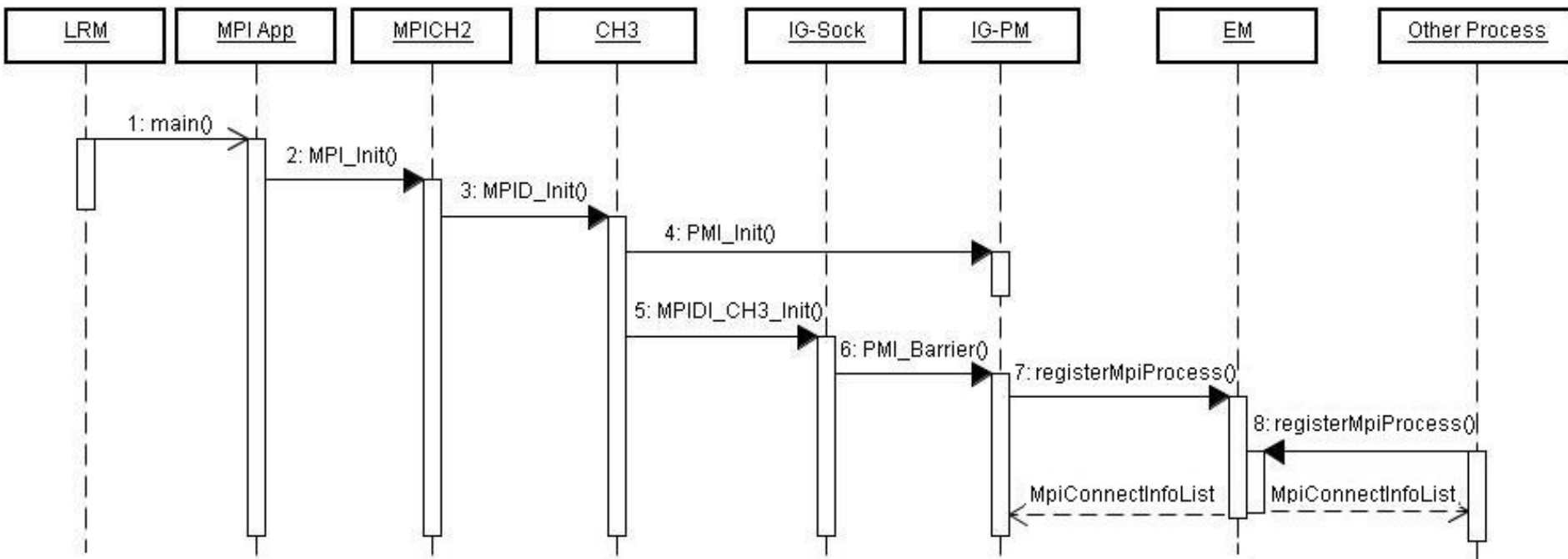
- replaces MPD
- get info from InteGrade to run MPI apps
- process location and synchronization
- management of checkpoints storage/retrieval

## IG-Sock

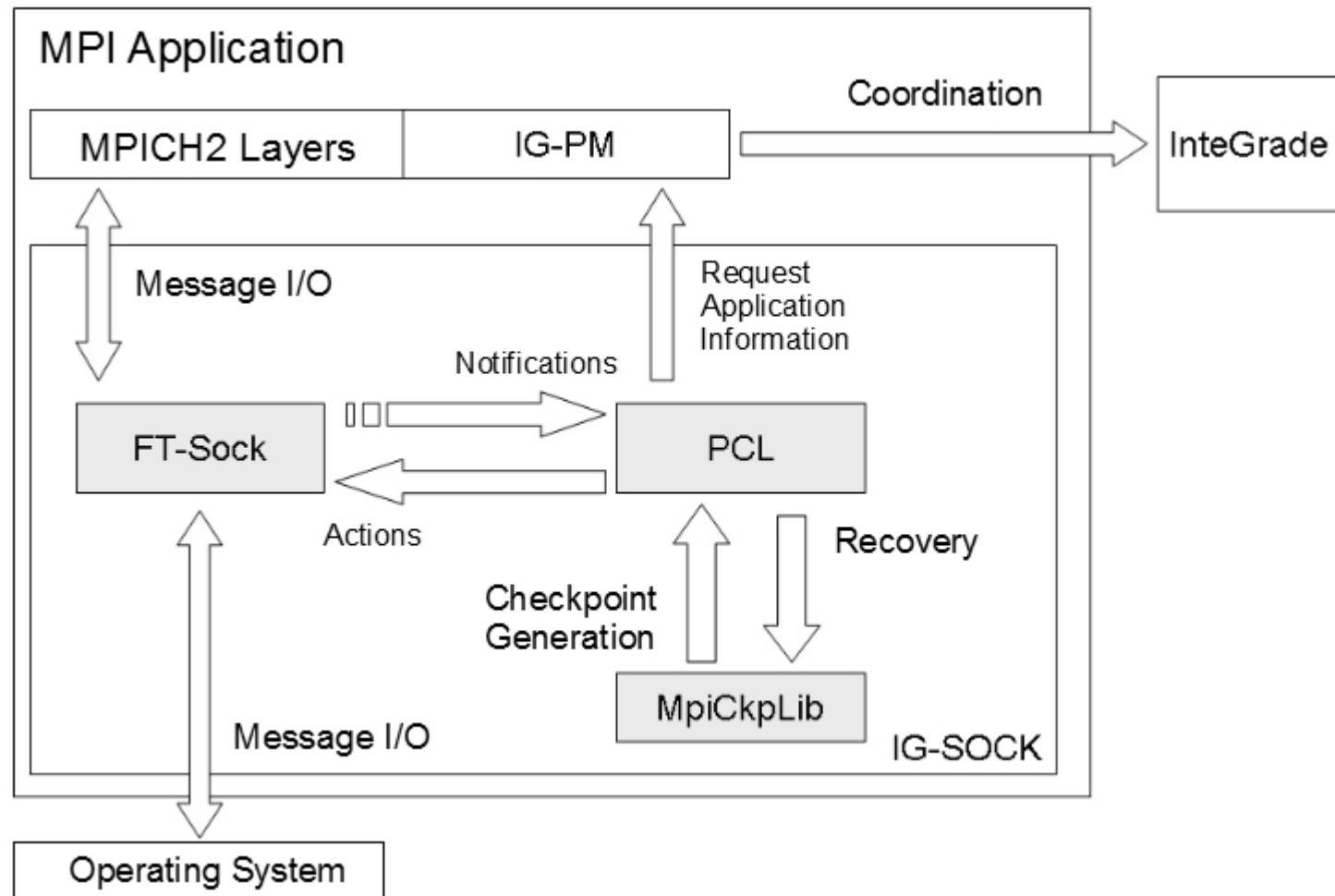
- replaces socket channel
- intercepts messages to generate checkpoints
- recovery from checkpoints



# Execution Protocol for MPI Apps

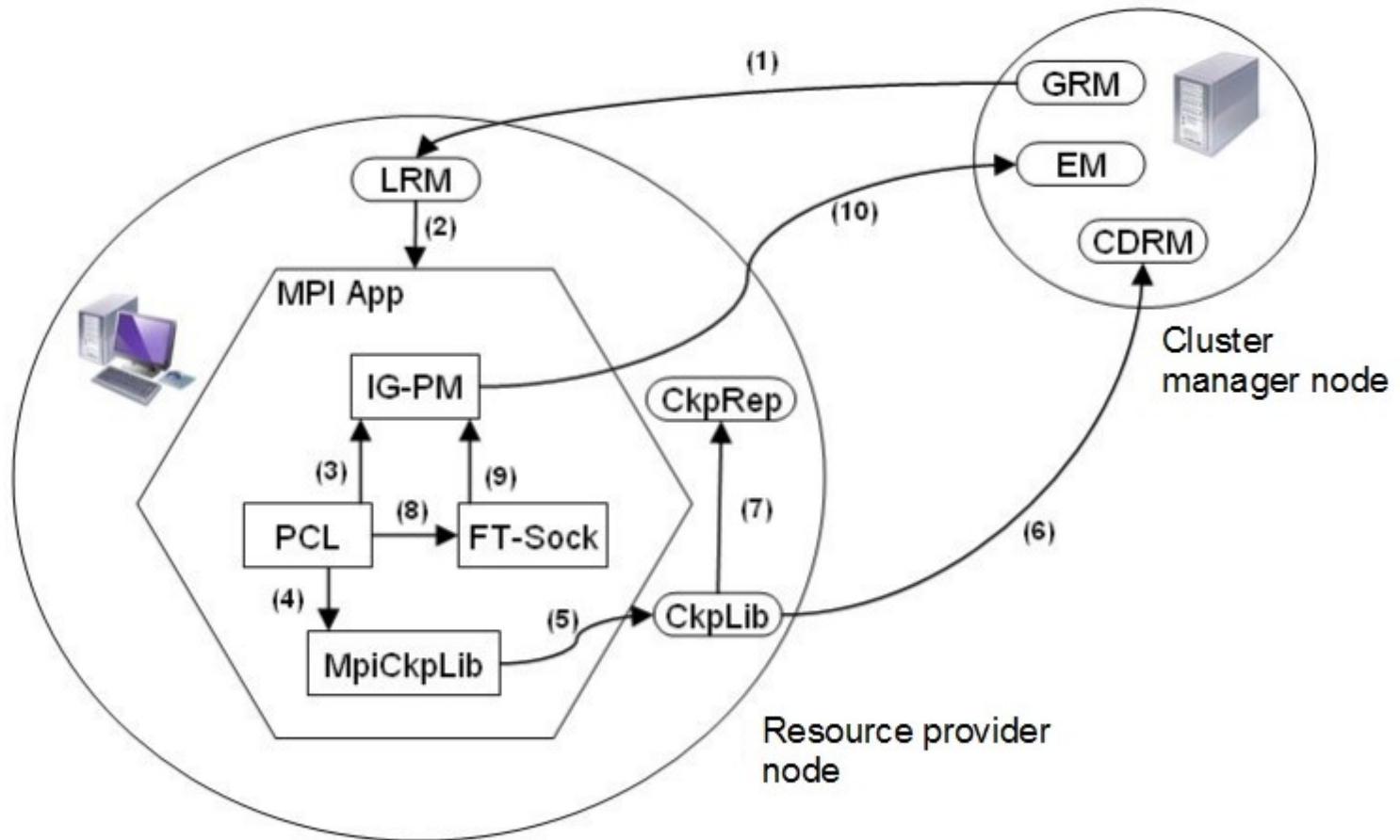


# IG-Sock Architecture

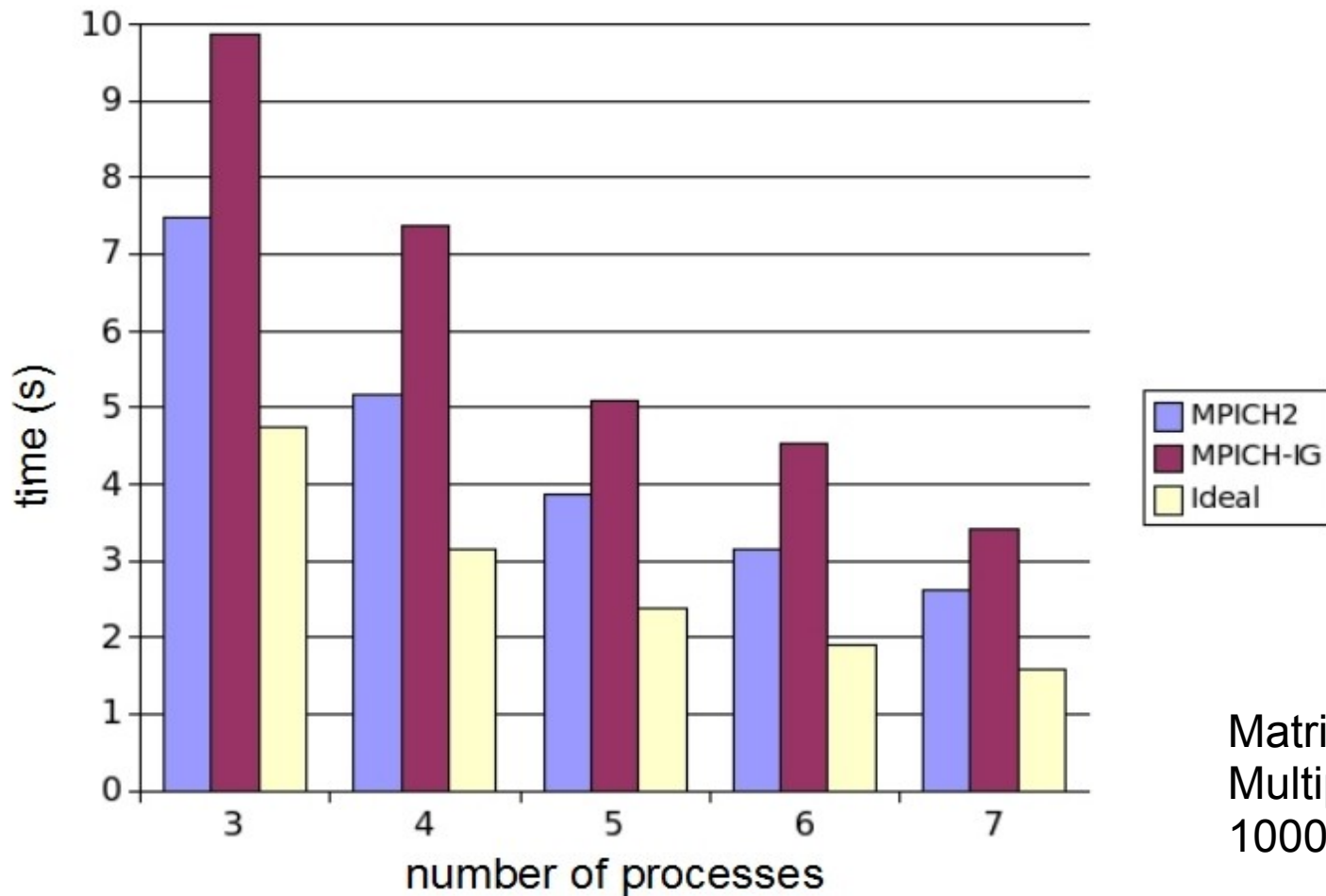




# Recovery Protocol



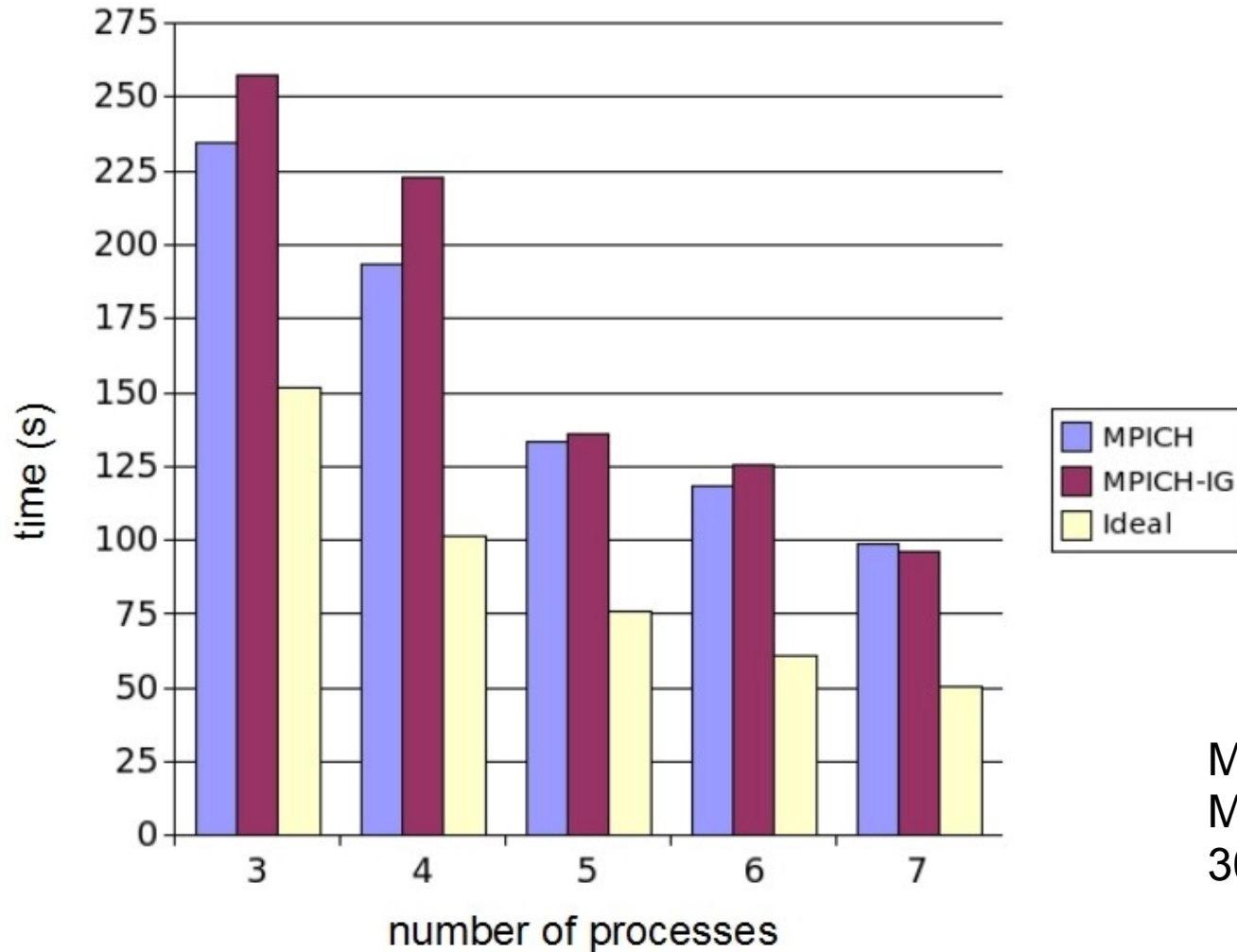
# Overall Performance Evaluation



Matrix  
Multiplication  
1000x1000



# Overall Performance Evaluation



Matrix  
Multiplication  
3000x3000



# Related Work

## MPICH-G2

- MPICH on GT4
- same overall approach, but w/out recovery

## MPICH-GF

- MPICH-G2 with checkpointing and recovery
- limited portability

## MPICH-V

- fault-tolerant MPI, but limited to homogeneous clusters



# Final Remarks

Transparent support for legacy MPI apps

- at source code level (needs recompilation)

Checkpointing & recovery

- enables a level of fault tolerance, essential for long-lived apps

Modular architecture

- possible to change components, e.g., communications channel and checkpointing strategy



# Future Work

Non-blocking checkpointing for performance

Support for the dynamic creation (spawning) of MPI processes

- requires an extension of InteGrade: scheduling of dynamic processes

Implement other kinds of channels: e.g., CORBA for interoperability

More detailed performance evaluation: e.g., with checkpointing



Thank you!



Execute application X

Execution descriptor: Load save

Name:

Base path:

Binaries:

Constraints:

Preferences:

Application type:  Regular  BSP  MPI  Parametric

Number of tasks:

Arguments:

Force Copies to execute on different Nodes

**Input Files**

Add Remove

**Output Files**

stderr  stdout Other: Add Remove

Submit Cancel

