Euro-Par 2005
Lisboa, Portugal

Dynamic Reconfiguration of Grid-Aware Applications in ASSIST

M. Aldinucci
A. Petrocelli, E. Pistoletti, M. Torquati,
M. Vanneschi, L. Veraldi, C. Zoccolo

ISTI - CNR, Pisa, Italy
University of Pisa, Italy

Euro-Par 2005
Lisboa, Portugal

Dynamic Reconfiguration of Grid-Aware Applications in ASSIST

M. Aldinucci
A. Petrocelli, E. Pistoletti, M. Torquati,
M. Vanneschi, L. Veraldi, C. Zoccolo

ISTI - CNR, Pisa, Italy
University of Pisa, Italy

ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

Outline

- Motivating ...
  - high-level programming for the grid
  - application adaptivity for the grid
- ASSIST basics
- Adaptivity in ASSIST
  - mechanisms
  - autonomic QoS managers
- Demo & experiments
- Concluding remarks

Outline

- Motivating ...
  - high-level programming for the grid
  - application adaptivity for the grid
- ASSIST basics
- Adaptivity in ASSIST
  - mechanisms
  - autonomic QoS managers
- Demo & experiments
- Concluding remarks

The grid

*"... coordinated resource sharing and problem solving in **dynamic**, multi institutional virtual organizations."*

Foster, Anatomy

"1) coordinates resources that are not subject to centralized control ..."
*"2) ... using **standard**, open, general-purpose protocols and interfaces"*
*"3) ... to deliver nontrivial **qualities of service**."*

Foster, What is the Grid?

Did you see J. Fortes invited talk?

The grid

*"... coordinated resource sharing and problem solving in **dynamic**, multi institutional virtual organizations."*

Foster, Anatomy

"1) coordinates resources that are not subject to centralized control ..."
*"2) ... using **standard**, open, general-purpose protocols and interfaces"*
*"3) ... to deliver nontrivial **qualities of service**."*

Foster, What is the Grid?

Did you see J. Fortes invited talk?

Moreover, since this is not **Euro-Seq**, I assume applications we are focusing on should be **parallel** (and hopefully **high-performance**).

// progr. & the grid

- concurrency exploitation, concurrent activities set up, mapping/scheduling, communication/synchronization handling and data allocation, ...
- manage resources heterogeneity and unreliability, networks latency and bandwidth unsteadiness, resources topology and availability changes, firewalls, private networks, reservation and jobs schedulers, ...

// progr. & the grid

- concurrency exploitation, concurrent activities set up, mapping/scheduling, communication/synchronization handling and data allocation, ...
- manage resources heterogeneity and unreliability, networks latency and bandwidth unsteadiness, resources topology and availability changes, firewalls, private networks, reservation and jobs schedulers, ...

... and a non trivial QoS for **applications**

// progr. & the grid

- concurrency exploitation, concurrent activities set up, mapping/scheduling, communication/synchronization handling and data allocation, ...
- manage resources heterogeneity and unreliability, networks latency and bandwidth unsteadiness, resources topology and availability changes, firewalls, private networks, reservation and jobs schedulers, ...

... and a non trivial QoS for **applications**

not easy leveraging only on middleware

Outline

- Motivating ...
  - high-level programming for the grid
  - application adaptivity for the grid
- ASSIST basics
- Adaptivity in ASSIST
  - mechanisms
  - autonomic QoS managers
- Demo & experiments
- Concluding remarks

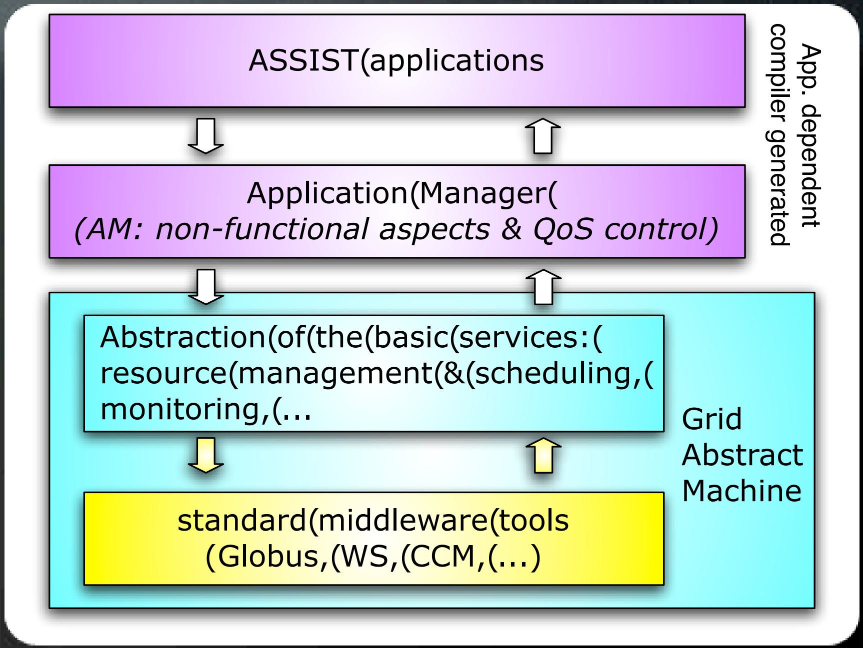
Outline

- Motivating ...
  - high-level programming for the grid
  - application adaptivity for the grid
- ASSIST basics
- Adaptivity in ASSIST
  - mechanisms
  - autonomic QoS managers
- Demo & experiments
- Concluding remarks

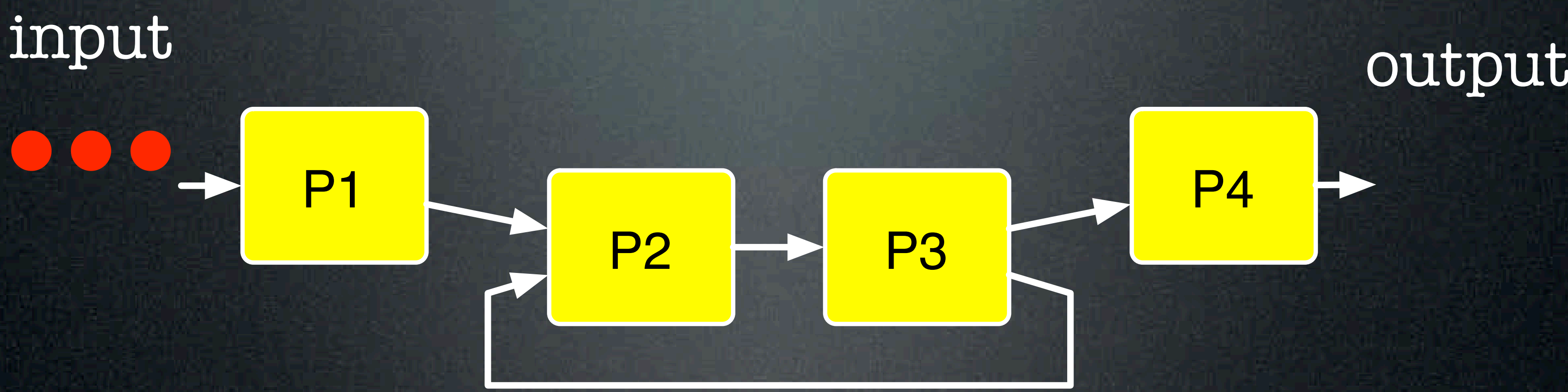
ASSIST idea

ASSIST is a high-level programming environment for grid-aware // applications. Developed at Uni. Pisa within several national/EU projects. First version in 2001. Open source under GPL.

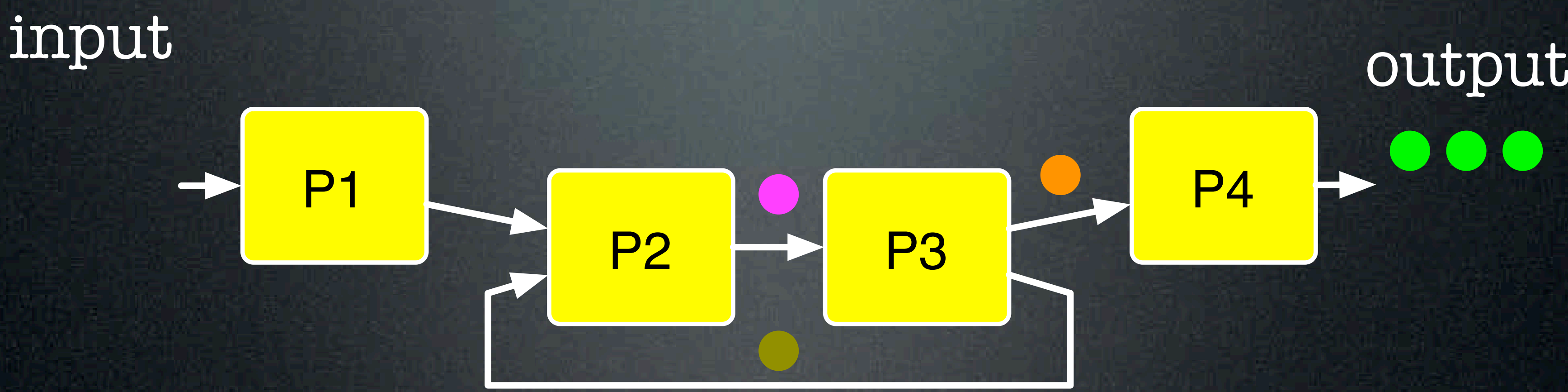
"moving most of the Grid specific efforts needed while developing high-performance Grid applications from programmers to grid tools and run-time systems"



app = graph of modules

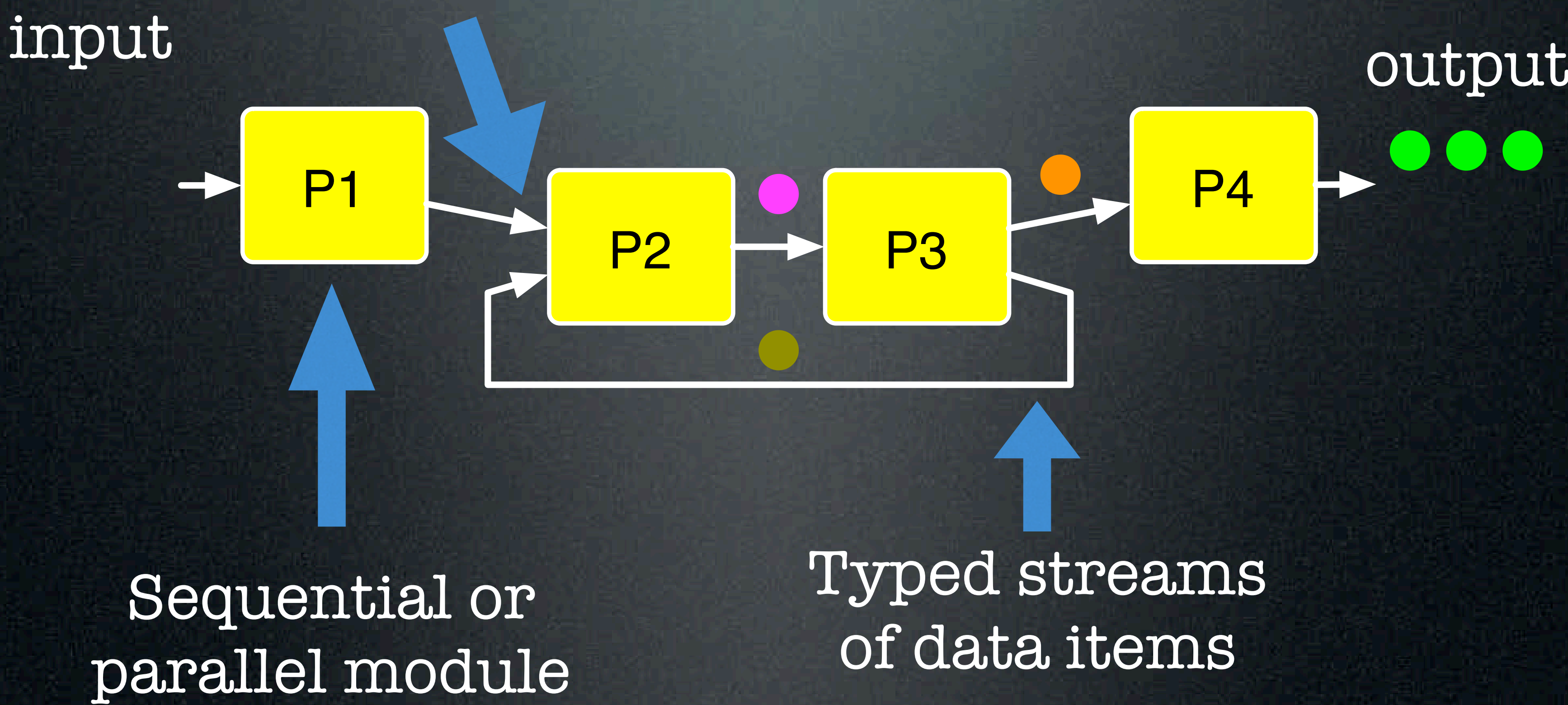


app = graph of modules

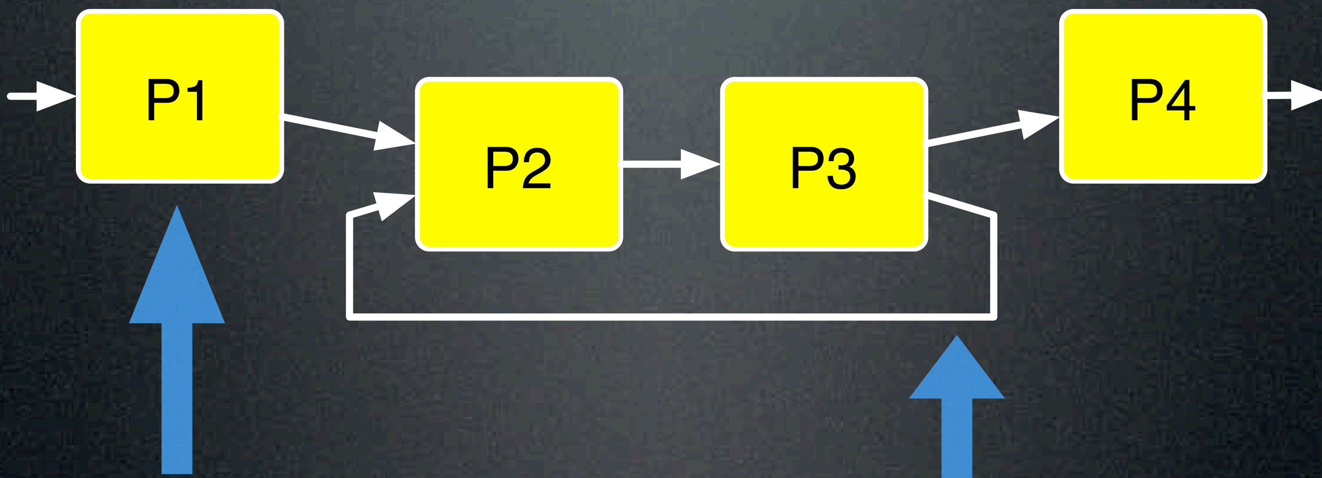


app = graph of modules

Programmable, possibly nondeterministic input behaviour



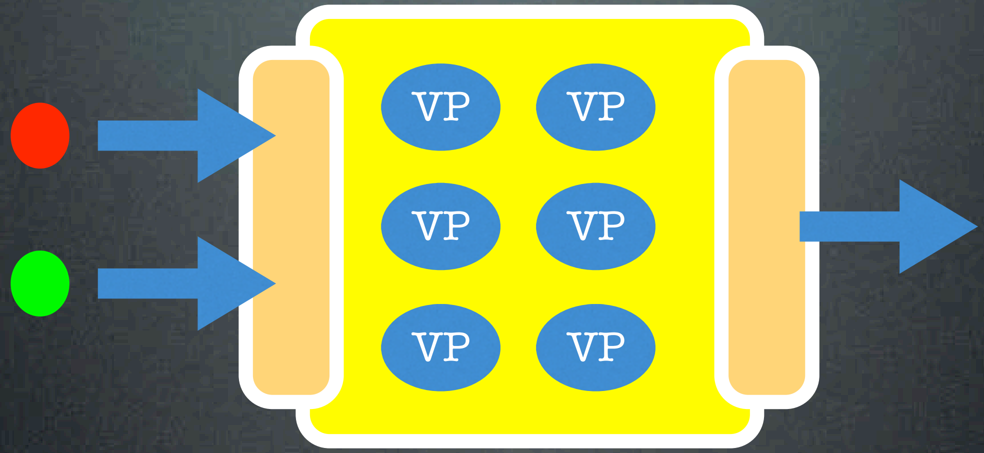
native + standard



ASSIST native or wrap
(MPI, CORBA, CCM, WS)

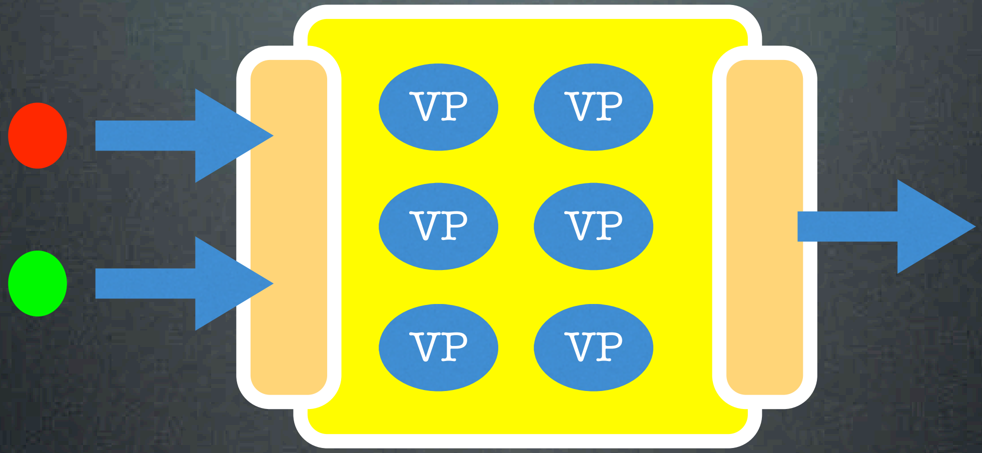
TCP/IP, Globus,
IIOP CORBA,
HTTP/SOAP

ASSIST native parmod

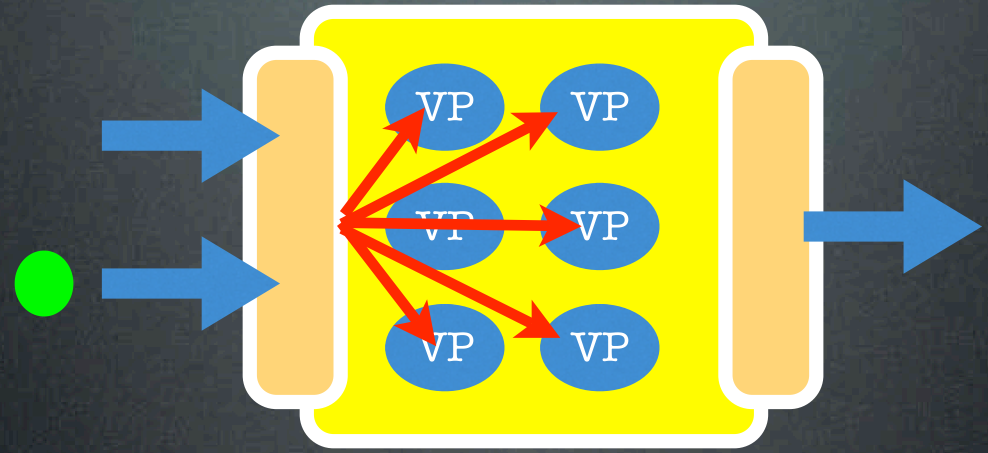


ASSIST native parmod

An “input section” can be programmed in a CSP-like way



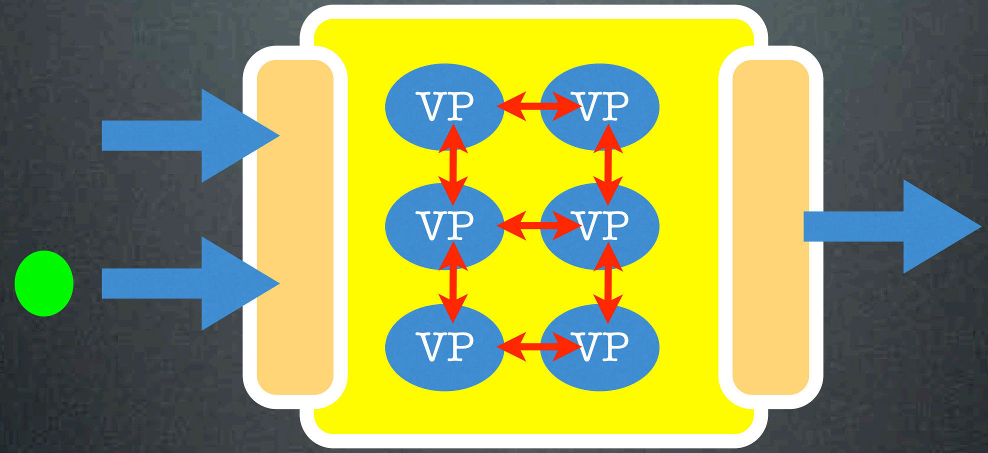
ASSIST native parmod



An "input section" can be programmed in a CSP-like way

Data items can be distributed (scattered, broadcasted, multicasted) to a set of **Virtual Processes** which are named accordingly to a topology

ASSIST native parmod



An "input section" can be programmed in a CSP-like way

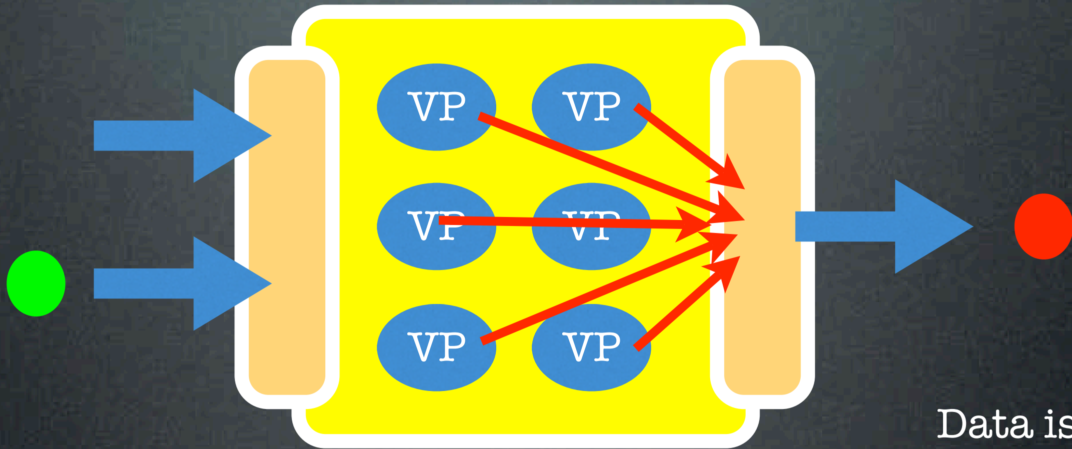
Data items can be distributed (scattered, broadcasted, multicasted) to a set of **Virtual Processes** which are named accordingly to a topology

Data items partitions are elaborated by VPs, possibly in iterative way

```
while(...)
  forall VP(in, out)
  barrier
```

data is logically shared by VPs (owner-computes)

ASSIST native parmod



An "input section" can be programmed in a CSP-like way

Data items can be distributed (scattered, broadcasted, multicasted) to a set of **Virtual Processes** which are named accordingly to a topology

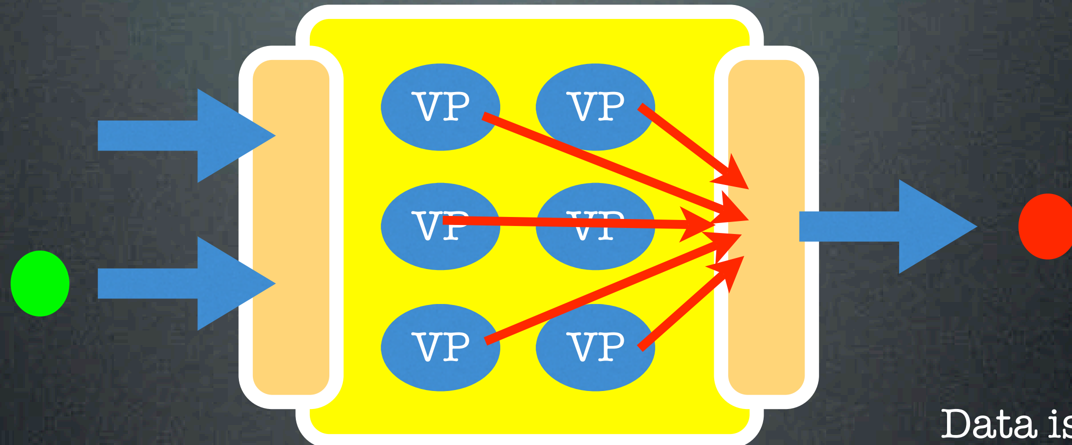
Data items partitions are elaborated by VPs, possibly in iterative way

```
while(...)
  forall VP(in, out)
  barrier
```

data is logically shared by VPs (owner-computes)

Data is eventually gathered accordingly to an user defined way

ASSIST native parmod



An “input section” can be programmed in a CSP-like way

Data is eventually gathered accordingly to an user defined way

Data items can be distributed (scattered, broadcasted, multicasted) to a set of **Virtual Processes** which are named accordingly to a topology

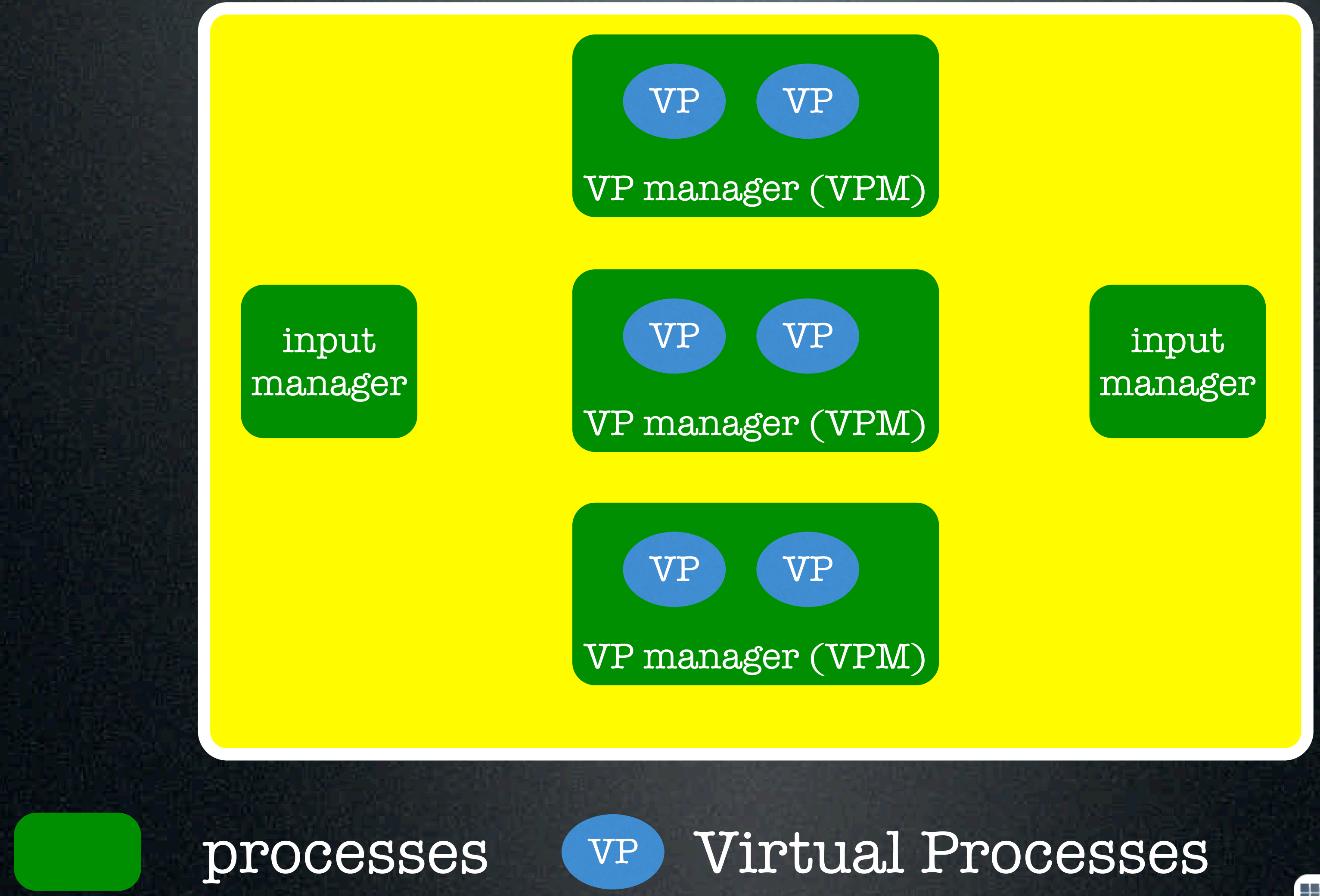
Data items partitions are elaborated by VPs, possibly in iterative way

```
while(...)
  forall VP(in, out)
  barrier
```

data is logically shared by VPs (owner-computes)

Easy to express standard paradigms (skeltons), such as **farm, deal, haloswap, map, apply-to-all, forall**, ...

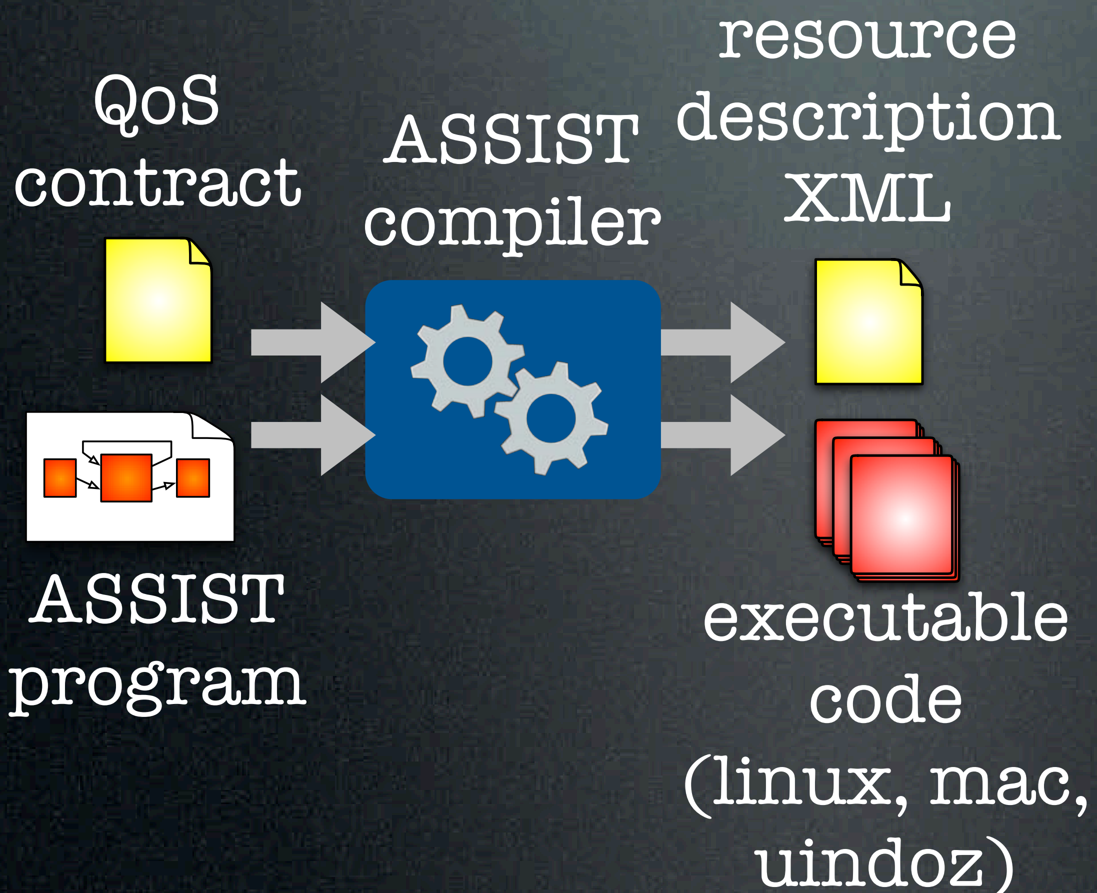
parmod implementation



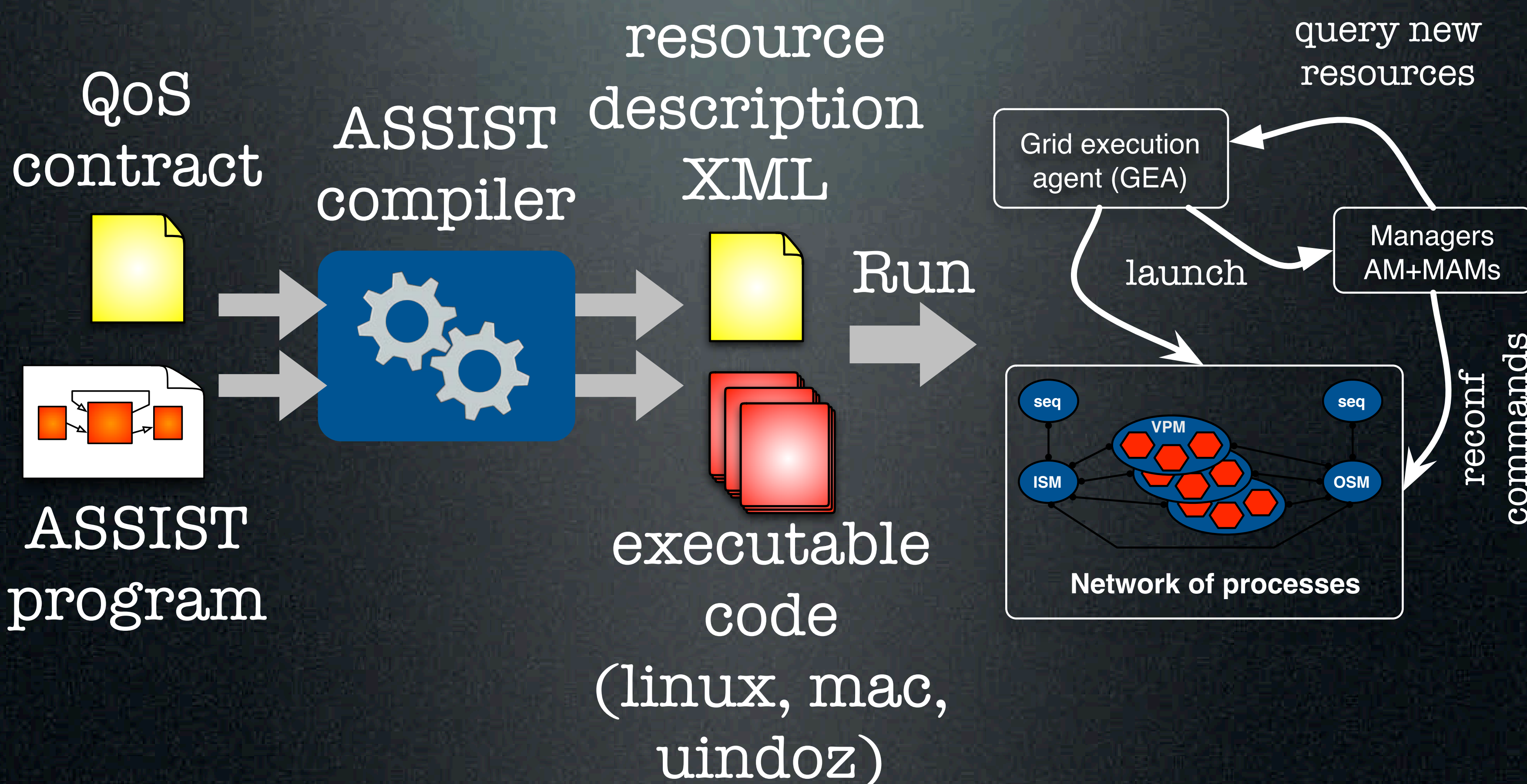
Compiling & Running

ASSIST
compiler

Compiling & Running

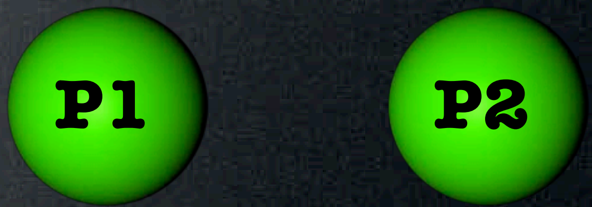


Compiling & Running



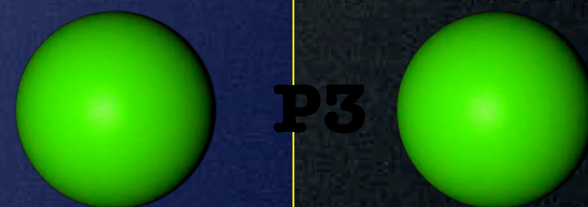
Outline

- Motivating ...
  - high-level programming for the grid
  - application adaptivity for the grid
- ASSIST basics
- Adaptivity in ASSIST
  - mechanisms (+ demo)
  - autonomic QoS managers
- Demo & experiments
- Concluding remarks



Outline

- Motivating ...
  - high-level programming for the grid
  - application adaptivity for the grid
- ASSIST basics
- Adaptivity in ASSIST
  - mechanisms (+ demo)
  - autonomic QoS managers



- Demo & experiments
- Concluding remarks

Application adaptivity

- Adaptivity aims to dynamically control program configuration (e.g. parallel degree) and mapping
  - for performance (high-performance is a natural sub-target)
  - for fault-tolerance (enable to cope with unsteadiness of resources, and some kind of faults)

Ingredients for the adaptivity recipe

Ingredients for the adaptivity recipe

1. Mechanism for adaptivity
   - reconf-safe points
     - in which points a parallel code can be safely reconfigured?
   - reconf-safe point consensus
     - different parallel activities may not proceed in lock-step fashion
   - add/remove/migrate computation & data

Ingredients for the adaptivity recipe

1. Mechanism for adaptivity
   - reconf-safe points
     - in which points a parallel code can be safely reconfigured?
   - reconf-safe point consensus
     - different parallel activities may not proceed in lock-step fashion
   - add/remove/migrate computation & data
2. Managing adaptivity
   - QoS contracts
     - Describing high-level QoS requirement for modules/applications
   - “self-optimizing” module
     - under the control of an autonomic manager

reconf-safe points

- In which points of the code the execution can be reconfigured?
  - low-level approach
    - the programmer places in the code calls to a suitable API, e.g. safe_point();
    - error-prone, time-consuming
  - ASSIST
    - automatically generated by the compiler, driven by program semantics
    - no artifactual synchronization added, already existing synchronizations are rather instrumented
    - overhead w.r.t. not adaptive code < 0.04%

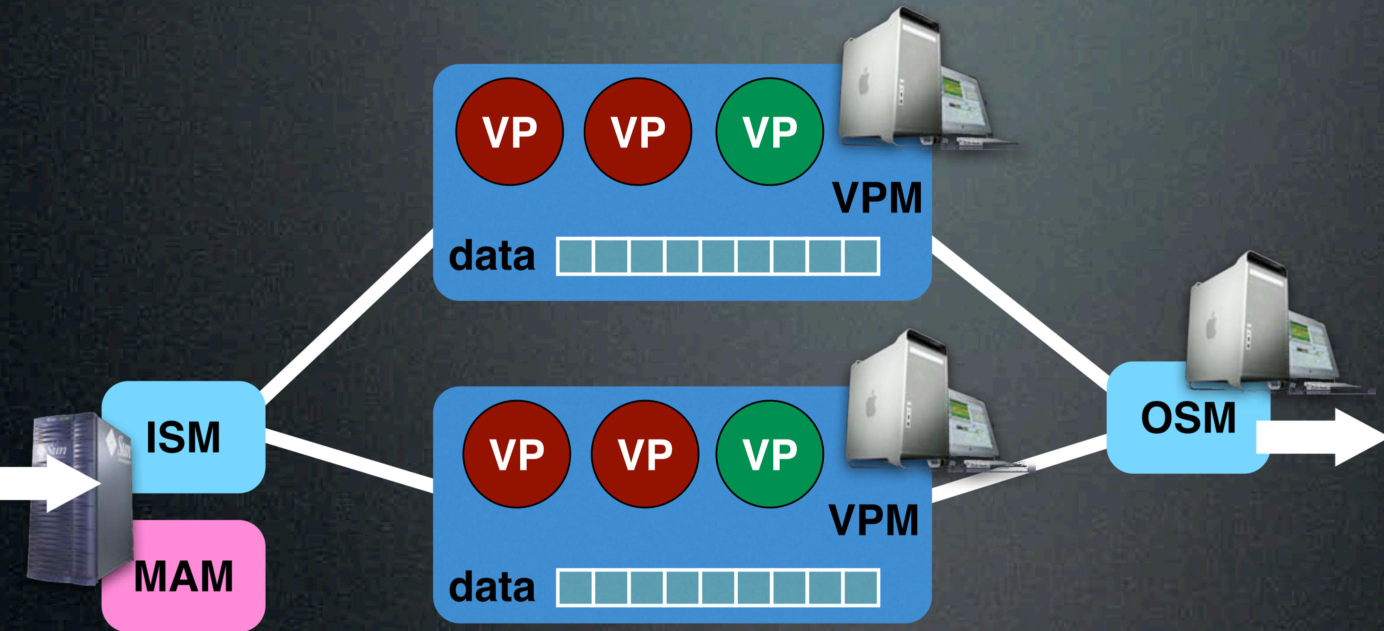
Distributed agreement

- The program reconfiguration actually starts only when all interested entities are ready to react
  - i.e. all processes have reached a suitable reconf-safe point
  - they agreed on which one
  - fresh resources are up and running
- distributed protocol

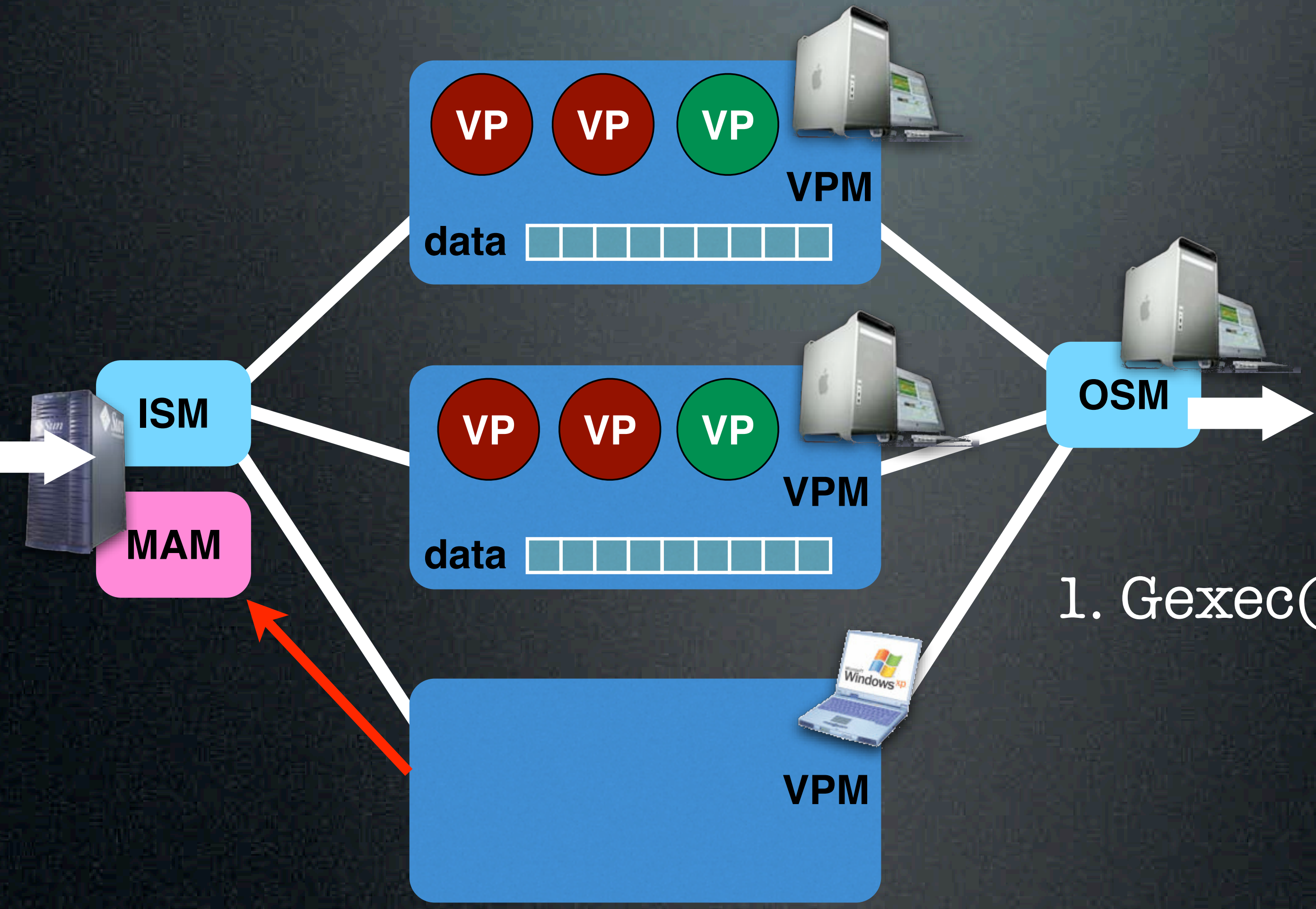
Basic operations

- Change parallelism degree
  - Add n VPMs to parmod
  - Remove n VPMs from a parmod
- Change mapping
  - Move k VPs from a VPM to another
  - Move a VPM from a PE to another
  - Dynamic load-balancing as sequence of migrate operations

Example: Add VPM

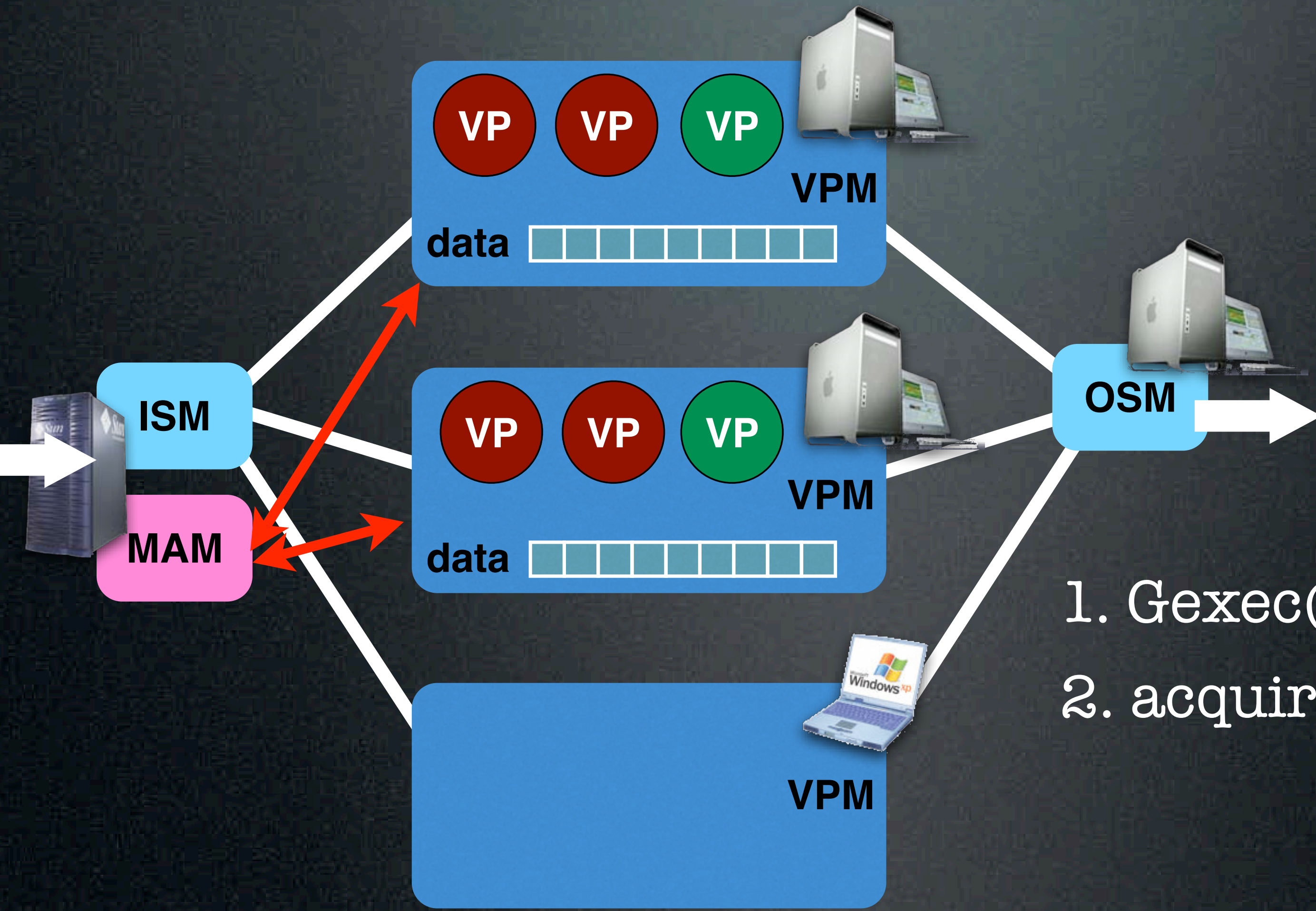


Example: Add VPM



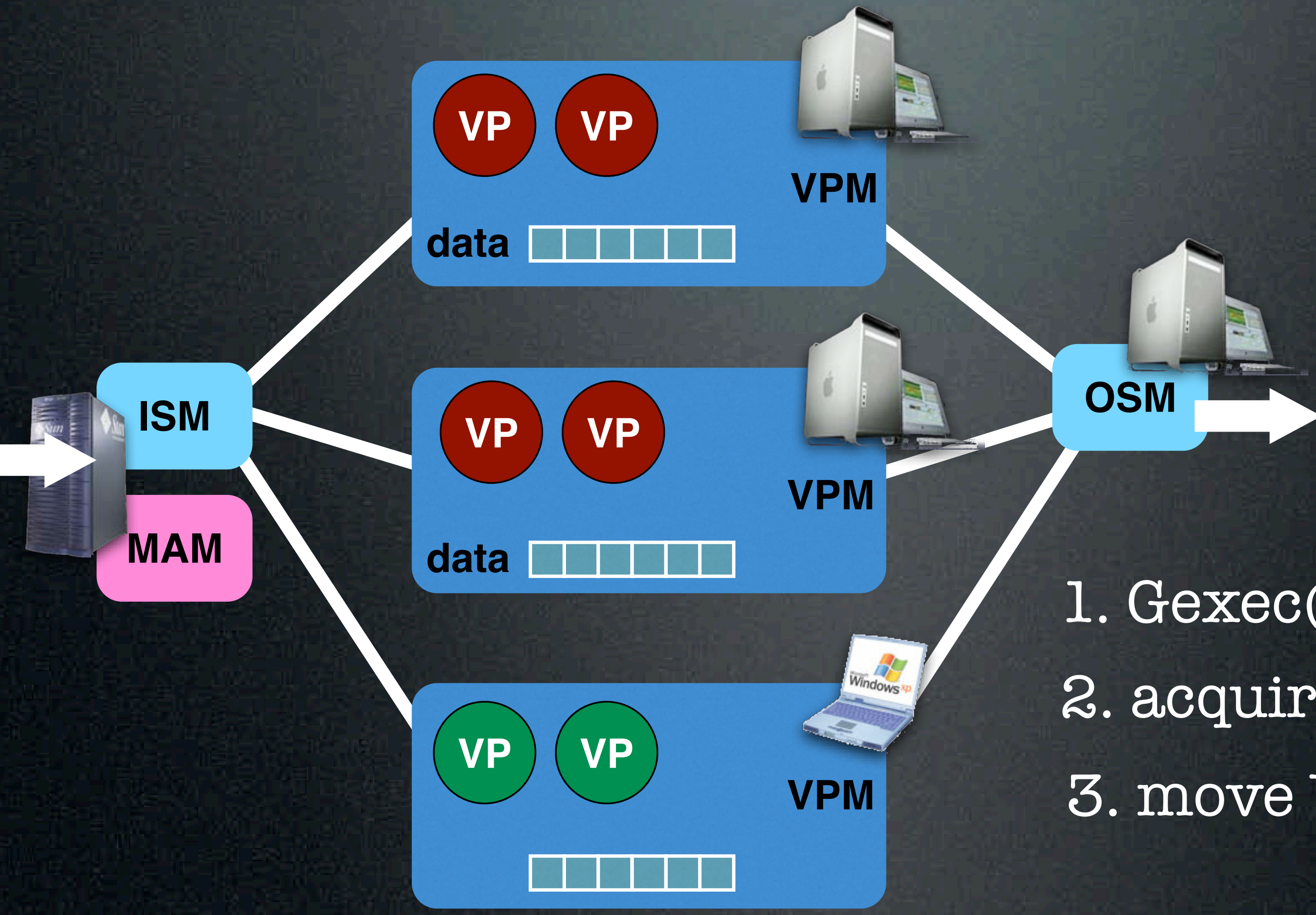
1. Gexec(newPE, VPM)

Example: Add VPM



1. Gexec(newPE, VPM)
2. acquire consensus

Example: Add VPM



1. Gexec(newPE, VPM)
2. acquire consensus
3. move VP and data

Only 3. is in the critical path

Overheads (milliseconds)

parmod kind	Data-parallel (with shared state)						Farm (without shared state)					
reconf. kind	add PEs			remove PEs			add PEs			remove PEs		
# of PEs involved	1→2	2→4	4→8	2→1	4→2	8→4	1→2	2→4	4→8	2→1	4→2	8→4
R_l on-barrier	1.2	1.6	2.3	0.8	1.4	3.7	–	–	–	–	–	–
R_l on-stream-item	4.7	12.0	33.9	3.9	6.5	19.1	~ 0	~ 0	~ 0	~ 0	~ 0	~ 0
R_t	24.4	30.5	36.6	21.2	35.3	43.5	24.0	32.7	48.6	17.1	21.6	31.9

GrADS papers reports overhead in the order of hundreds of seconds (K. Kennedy et al. 2004), this is mainly due to the stop/restart behavior, not to the different running env.

run begin
with 1 VPM

Demo Here !

reconf. console

run begin
with 1 VPM

lines arrives
slowly one
after the other
(par. degree=1)

4 fresh VPMs
are started

reconf: add the
4 VPMs

fresh VPMs are
added to the app.
lines arrives
much faster
(par. degree=5)

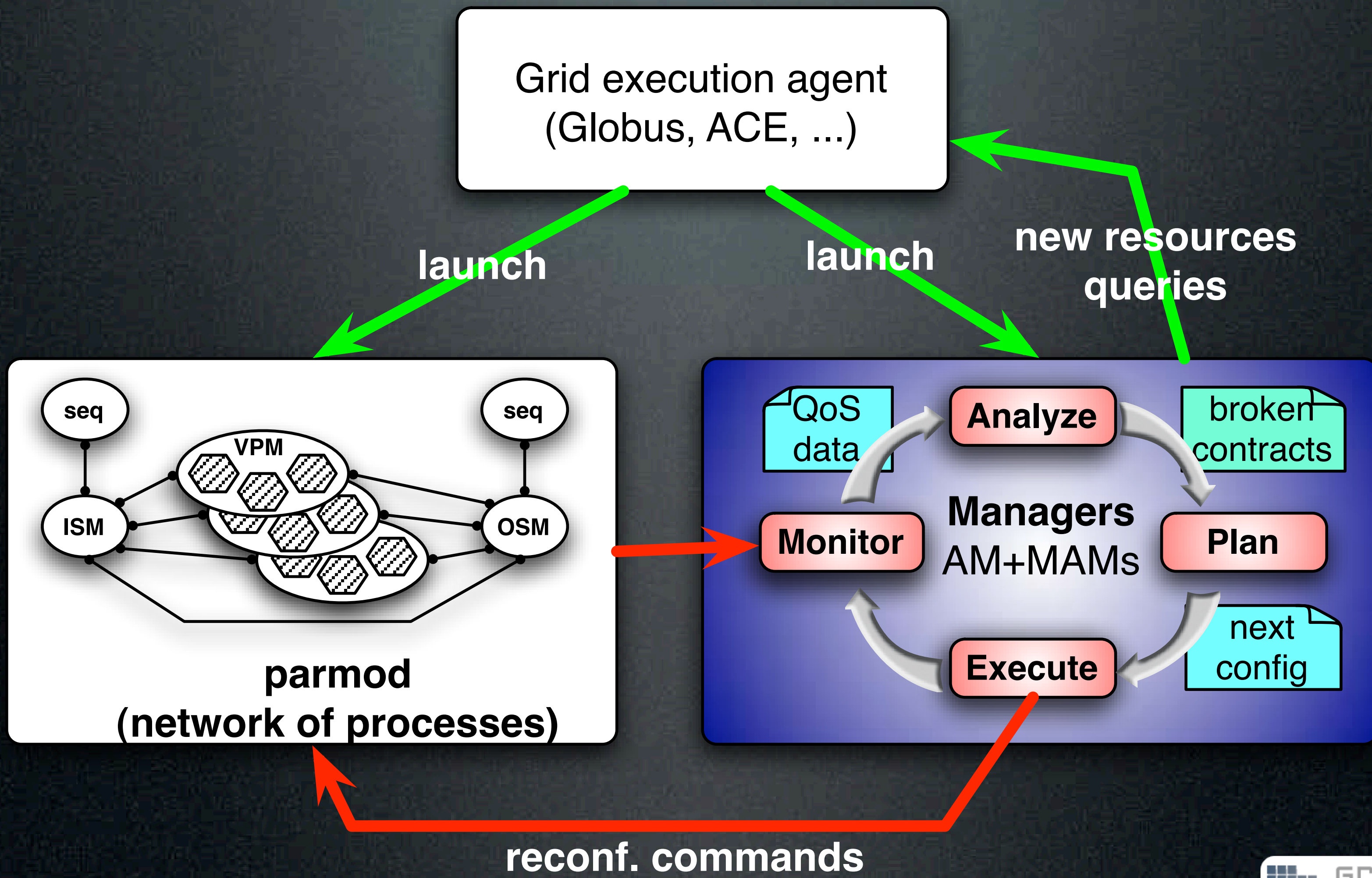
reconf: release
3 VPMs

lines arrives
a bit slower
(par. degree=2)

Demo Here !

reconf. console

Managing adaptivity



parmod autonomic manager

1. monitor
   - collect execution stats: machine load, VPM service time, input/output queues lenghts, ...
2. analyze
   - instanciate performance models with monitored data, detect broken contract, in and in the case try to indivituate the problem
3. plan
   - select a (predefined or user defined) strategy to reconvey the contract to valid status. The strategy is actually a list of mechanism to apply.
4. execute
   - leverage on mechanism to apply the plan

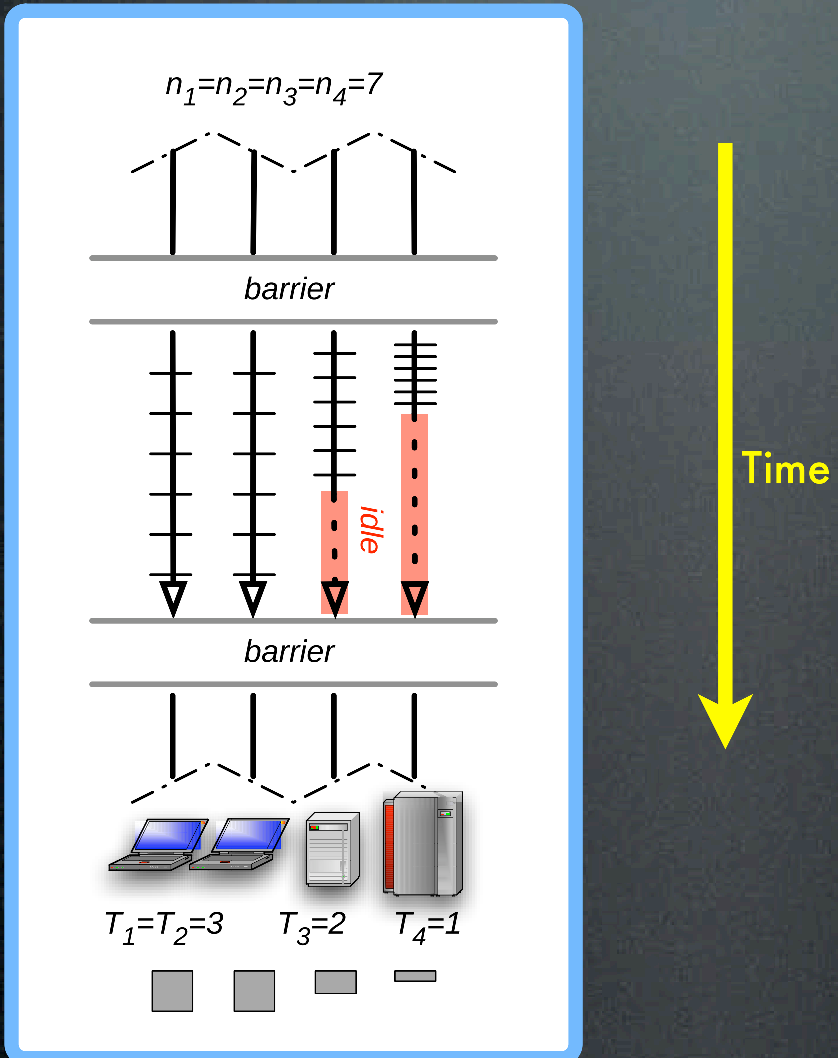
QoS contract

(of the experiment I'll show you in a minute)

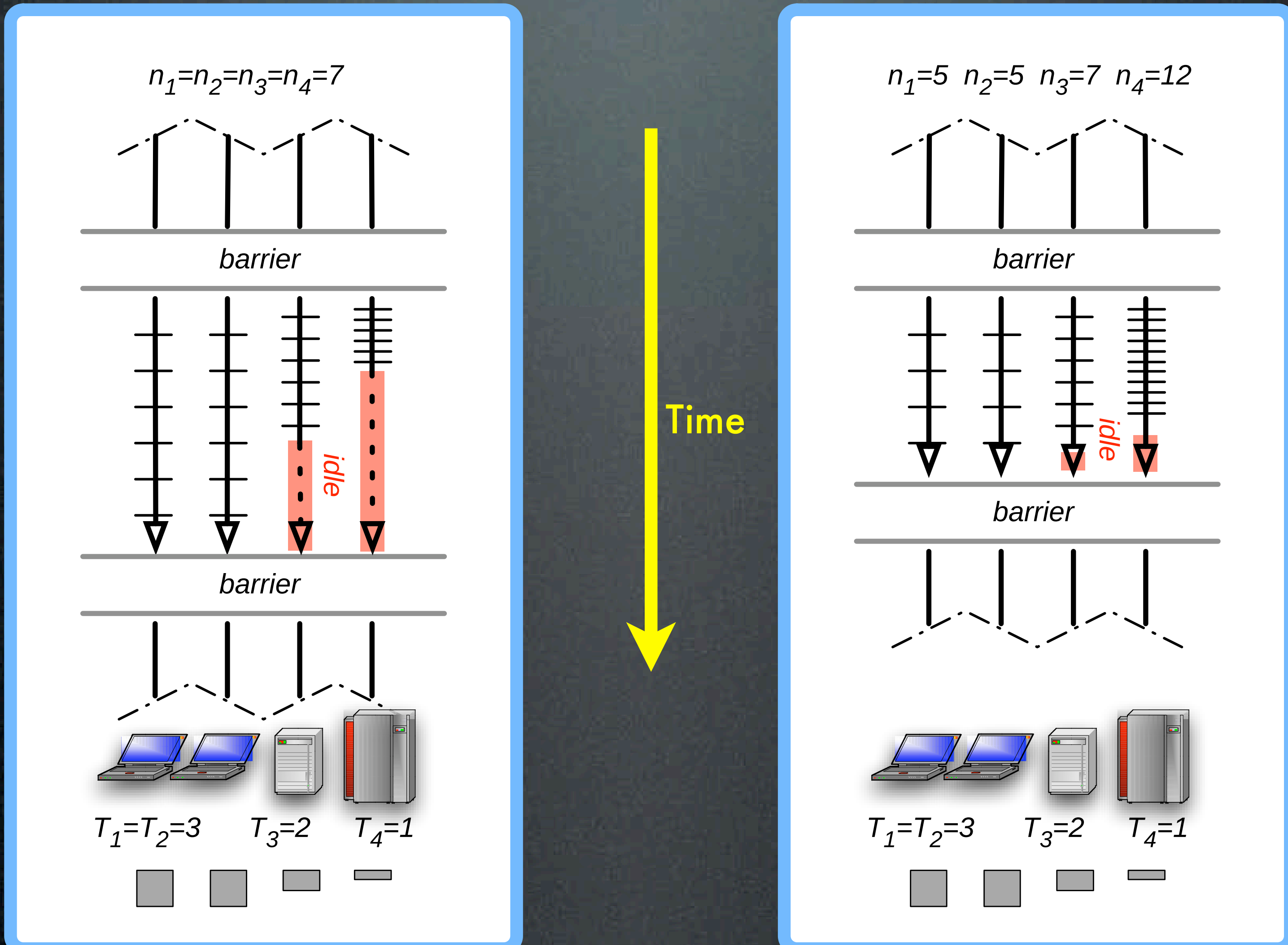
Perf. features	QL_i (input queue level), QL_o (input queue level), T_{ISM} (ISM service time), T_{OSM} (OSM service time), N_w (number of VPMs), $T_w[i]$ (VPM$_i$ avg. service time), T_p (parmod avg. service time)
Perf. model	$T_p = \max\{T_{ISM}, \sum_{i=1}^{n} T_w[i]/n, T_{OSM}\}$, $T_p < K$ (goal)
Deployment	arch = (i686-pc-linux-gnu $\vee$ powerpc-apple-darwin*)
Adapt. policy	goal_based

Performance models: an example (DP load balancing)

Performance models: an example (DP load balancing)



Performance models: an example (DP load balancing)



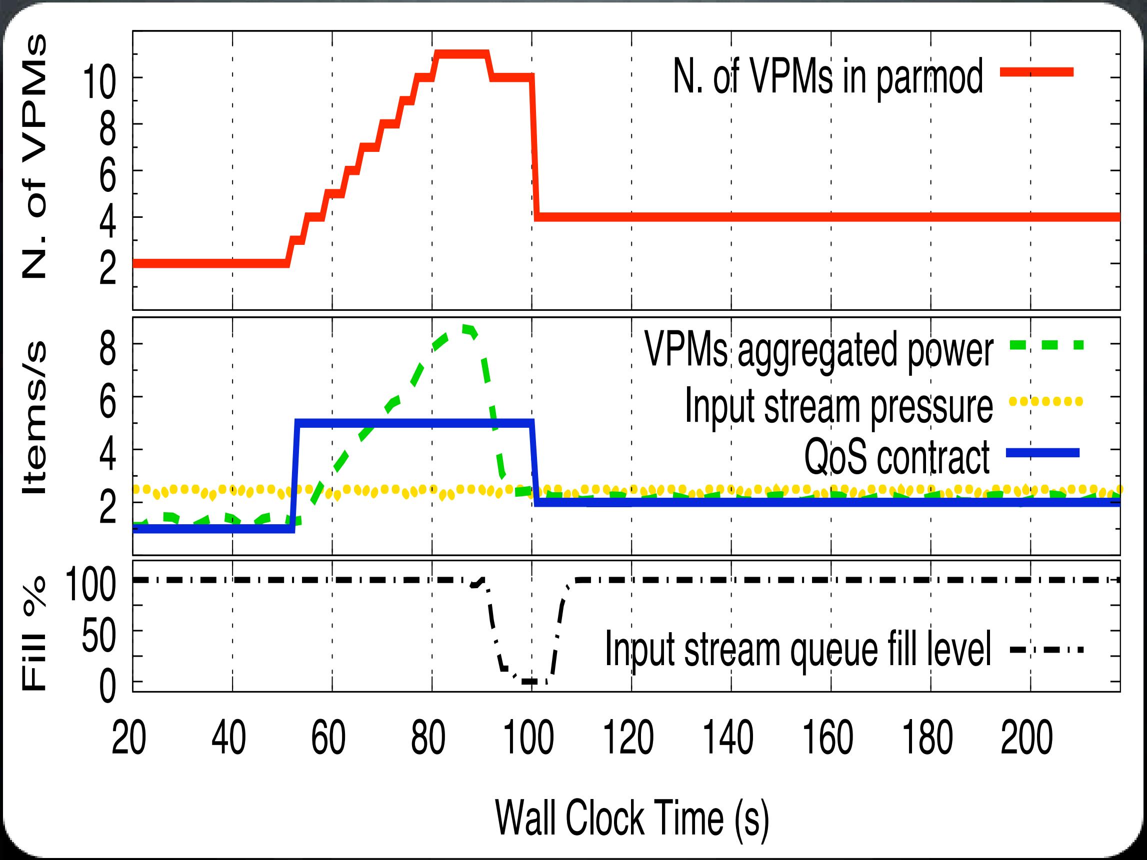
Outline

- Motivating ...
  - high-level programming for the grid
  - application adaptivity for the grid
- ASSIST basics
- Adaptivity in ASSIST
  - mechanisms (+ demo)
  - autonomic QoS managers
- Experiments
- Concluding remarks

Outline

- Motivating ...
  - high-level programming for the grid
  - application adaptivity for the grid
- ASSIST basics
- Adaptivity in ASSIST
  - mechanisms (+ demo)
  - autonomic QoS managers
- Experiments
- Concluding remarks

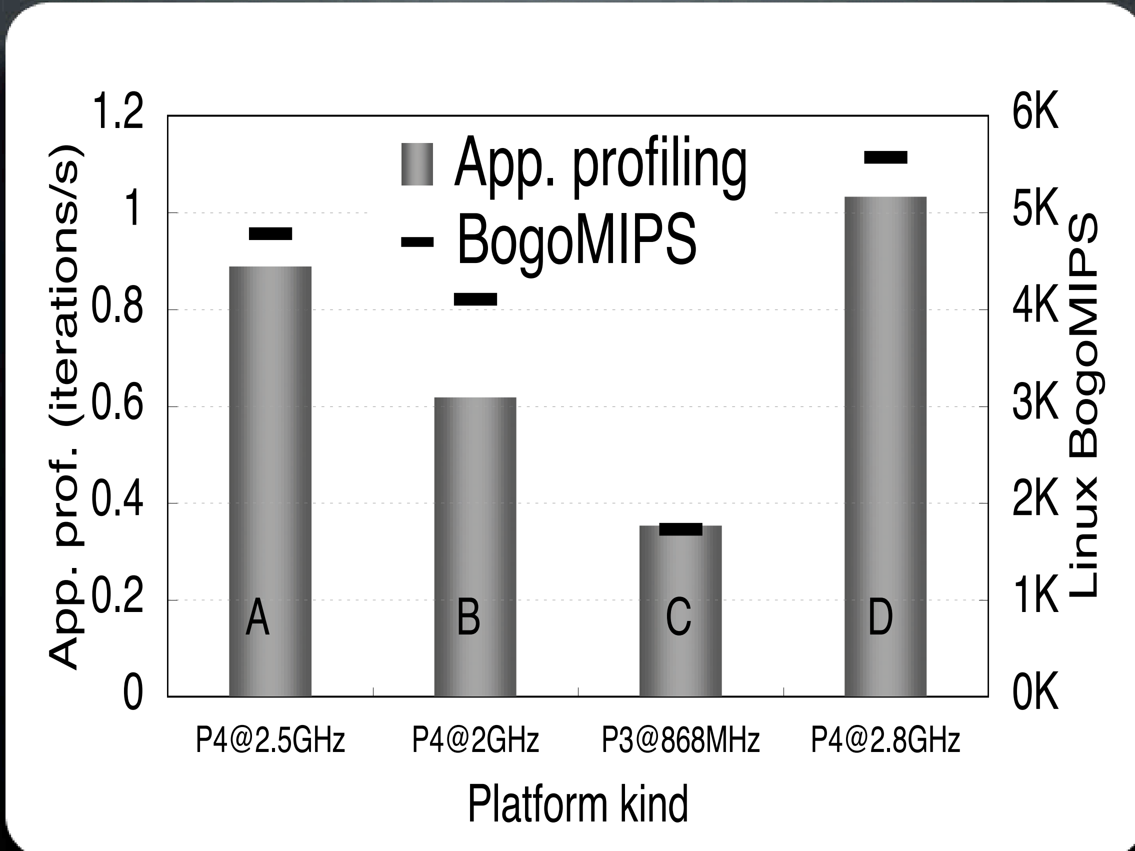
Perf(P1)
Perf(P2)
Perf(P3)
Perf(P4)



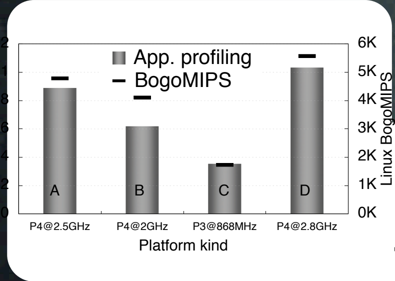
Farm:

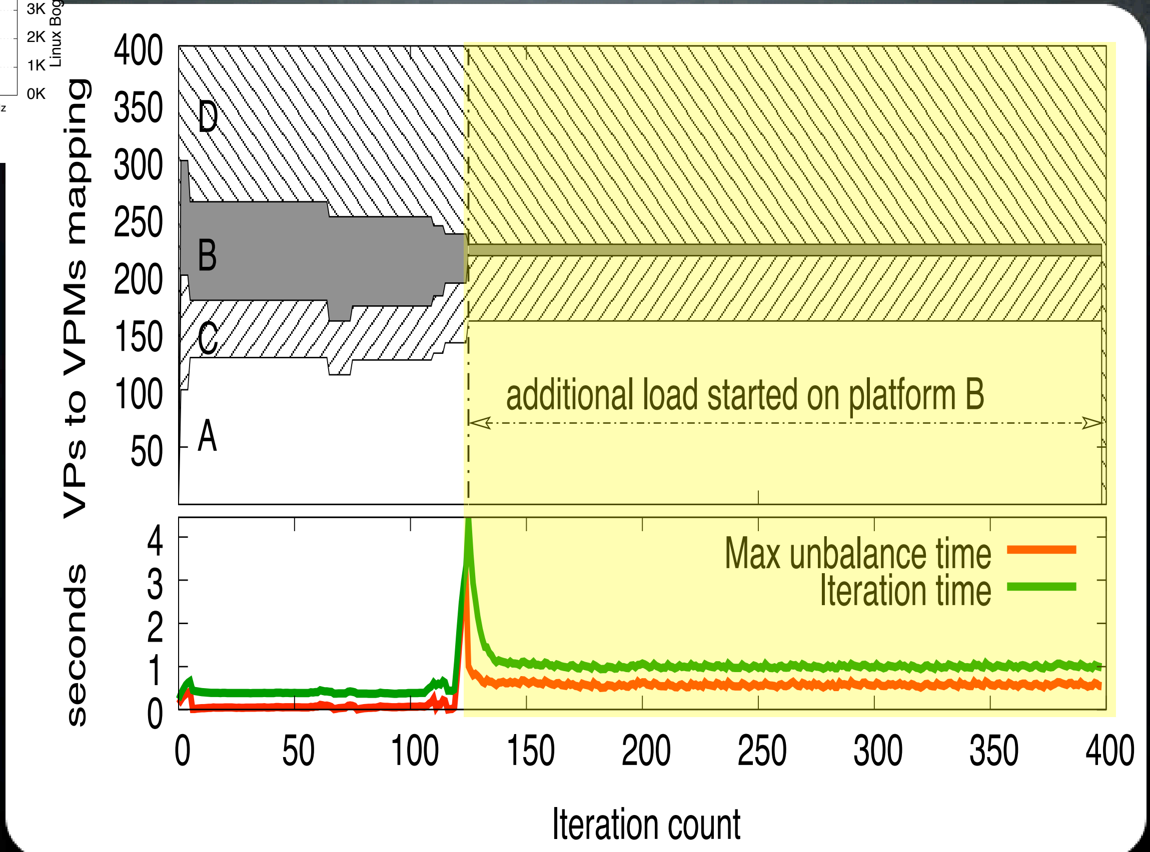
contract: keep a given service time

contract change along the run


App. profiling
BogoMIPS
App. prof. (iterations/s)
Linux BogoMIPS
1.2
1
0.8
0.6
0.4
0.2
0
6K
5K
4K
3K
2K
1K
0K
A
B
C
D
P4@2.5GHz
P4@2GHz
P3@868MHz
P4@2.8GHz
Platform kind

Running Env





Data parallel (shortest path)
Machine B externally overloaded after a while

Conclusions 1/2

- Application adaptivity in ASSIST
  - complex, but trasparent (no burden for the programmers)
    - they should just define they QoS requirements
    - perf. models are automatically generated from program structure (and don't depend on seq. funct.)
  - dynamically controlled, efficiently managed
    - catch both platforms unsteadiness and code irregular behavior in running time
    - performance models not critical, reconfiguration does not stop the application
    - key feature for the grid

Conclusions 2/2

Conclusions 2/2

- ASSIST cope with
  - grid platform unsteadiness
  - interoperability with standards
    - and rely on them for many features
  - high-performance
  - app deployment problems on grid
    - private networks, job schedulers, firewalls, ...

Conclusions 2/2

- ASSIST cope with
  - grid platform unsteadiness
  - interoperability with standards
    - and rely on them for many features
  - high-performance
  - app deployment problems on grid
    - private networks, job schedulers, firewalls, ...
- We currently working on
  - QoS of the whole application through hierarchy of managers
  - components, fault-tolerance, efficient launch time mapping
  - in cooperation with many coreGRID partners

Thank you

- ASSIST is open source under GPL, available on the web
- http://www.di.unipi.it/Assist.html
- or search with google:
  ASSIST programming environment