Edinburgh, March 16th 2005

Towards Grid-aware Program Adaptivity in ASSIST

M. Aldinucci
Italian National Research Council
& University of Pisa
www.di.unipi.it/~aldinuc




Skeletons

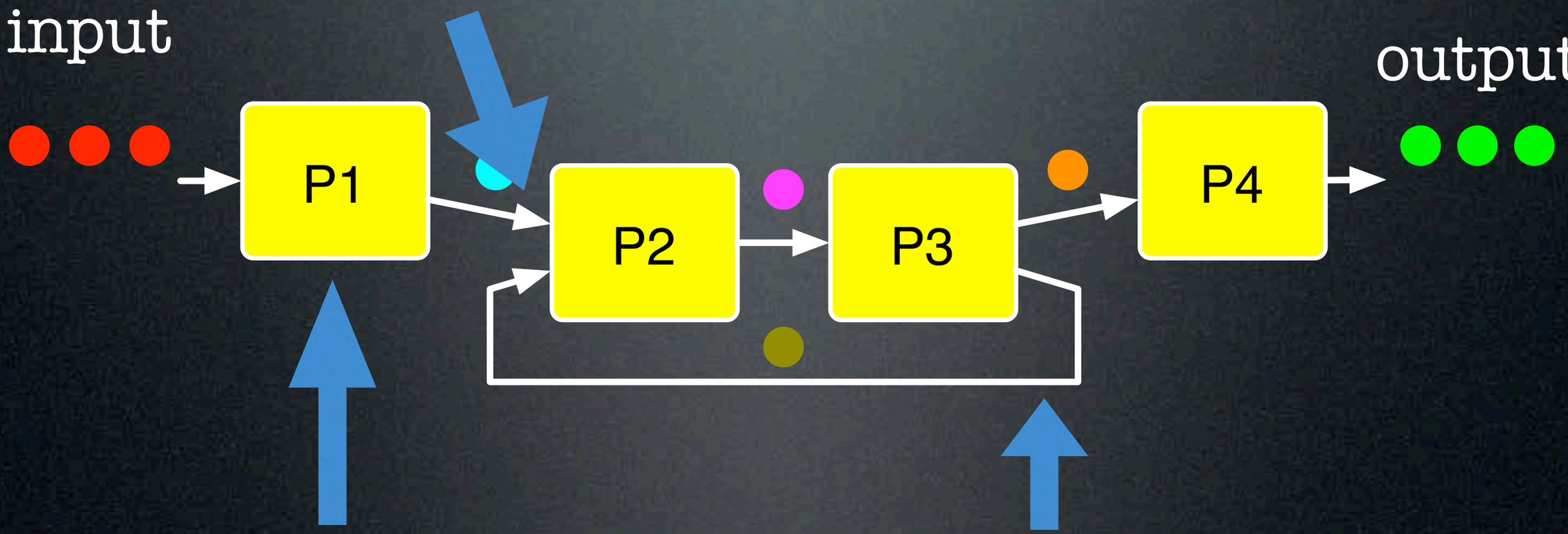
- First time here ... however, [Col89] and Prag. Manifesto appear in dozen of my papers
- Since ‘91 Pisa active in the field: P3L,
  - OCaml-P3L,
  - SkIE 1998(C+MPI),
  - Skel-BSP 2000, FAN 2000
  - Lithium 2001 (Java MacroDataFlow),
  - eskimo 2002 (Cilk-like DSM-based),
  - Muskel 2003 (Jaxta P2P), ...
- ... now ASSIST (since 2001)

Skeletons

- parallelism exploitation patterns (paradigms)
- It is clearly unrealistic assume that skeletons can provide all the parallelism we need ...
- Structured parallel programming build bridges to the programming standards of the day, refining or constraining only where strictly necessary. It should respect the conceptual model of these standards, offering skeletons as enhancement rather than as competition. We should construct our systems to allow the integration of skeletal and ad-hoc parallelism in a well-defined way ...

ASSIST in 5 minutes

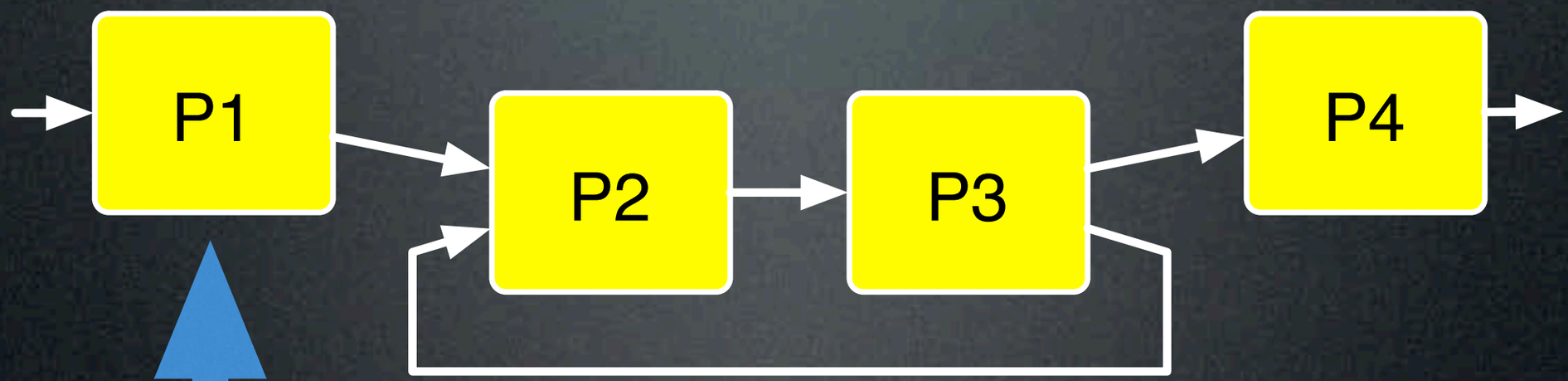
Programmable, possibly nondeterministic input behaviour



Sequential or parallel module

Typed streams of data items

ASSIST in 4 minutes

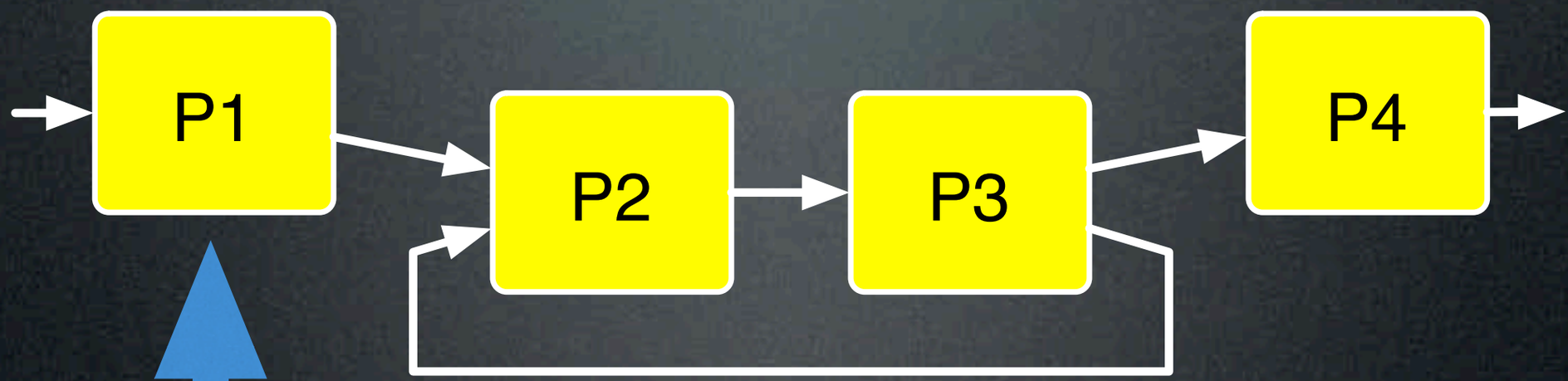


ASSIST native or wrap
(MPI, CORBA, CCM, WS)

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

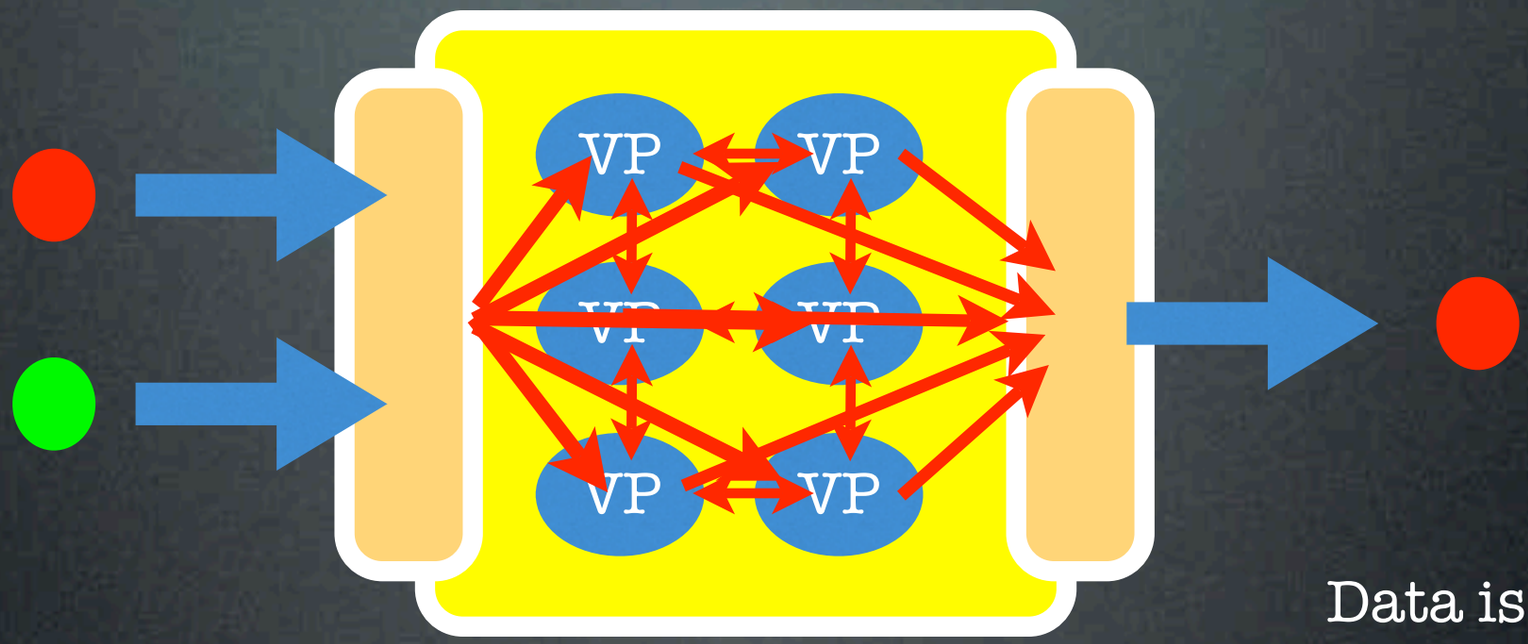
ASSIST in 3 minutes

ASSIST native parallel module
(aka parmod)



farm, deal, haloswap, map,
apply-to-all, forall, ...

ASSIST in 2 minutes



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

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

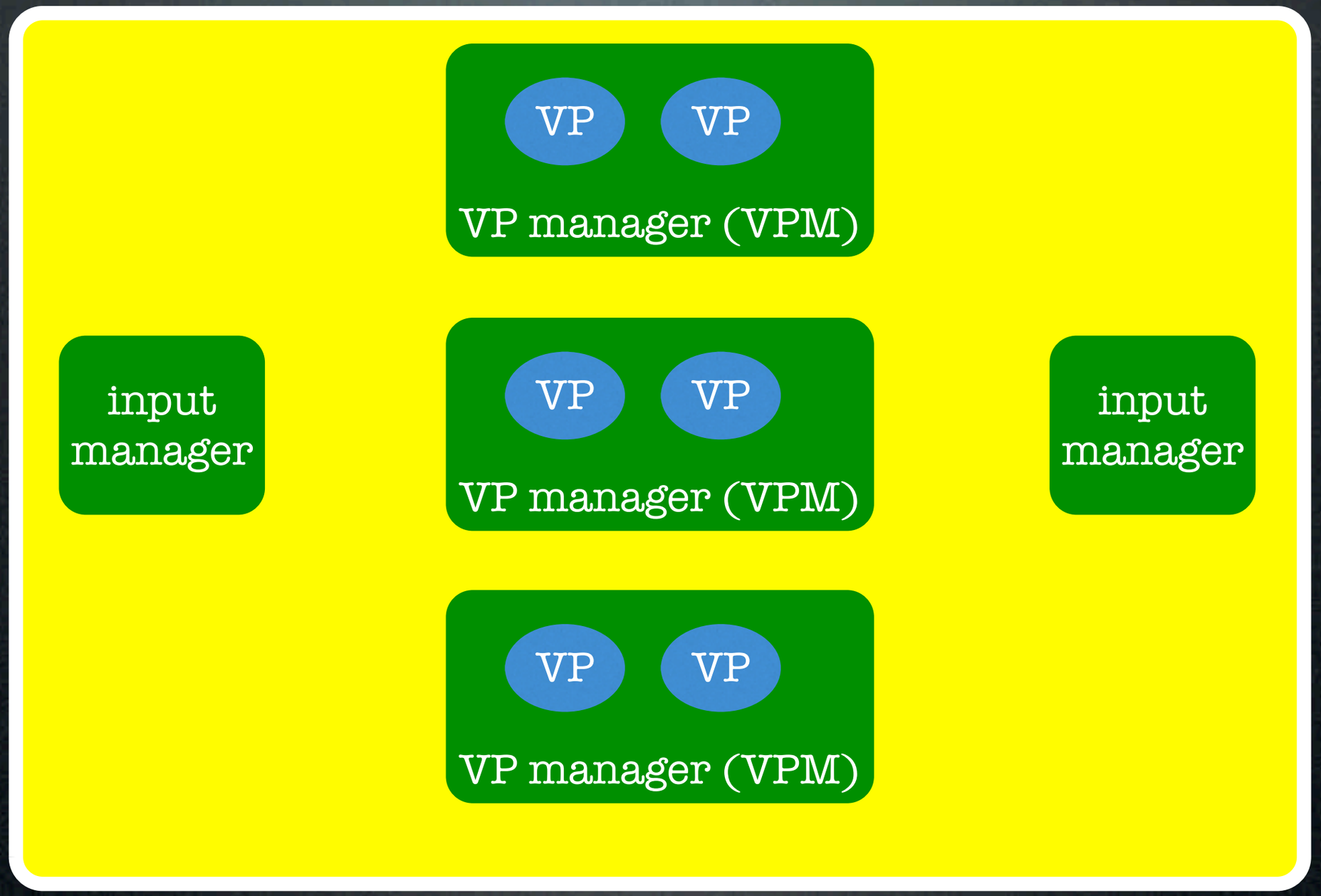
Data items partitions are elaborated by VPs, possibly in iterative way

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

data is logically shared by VPs (owner-computes)

Data is eventually gathered accordingly to an user defined way

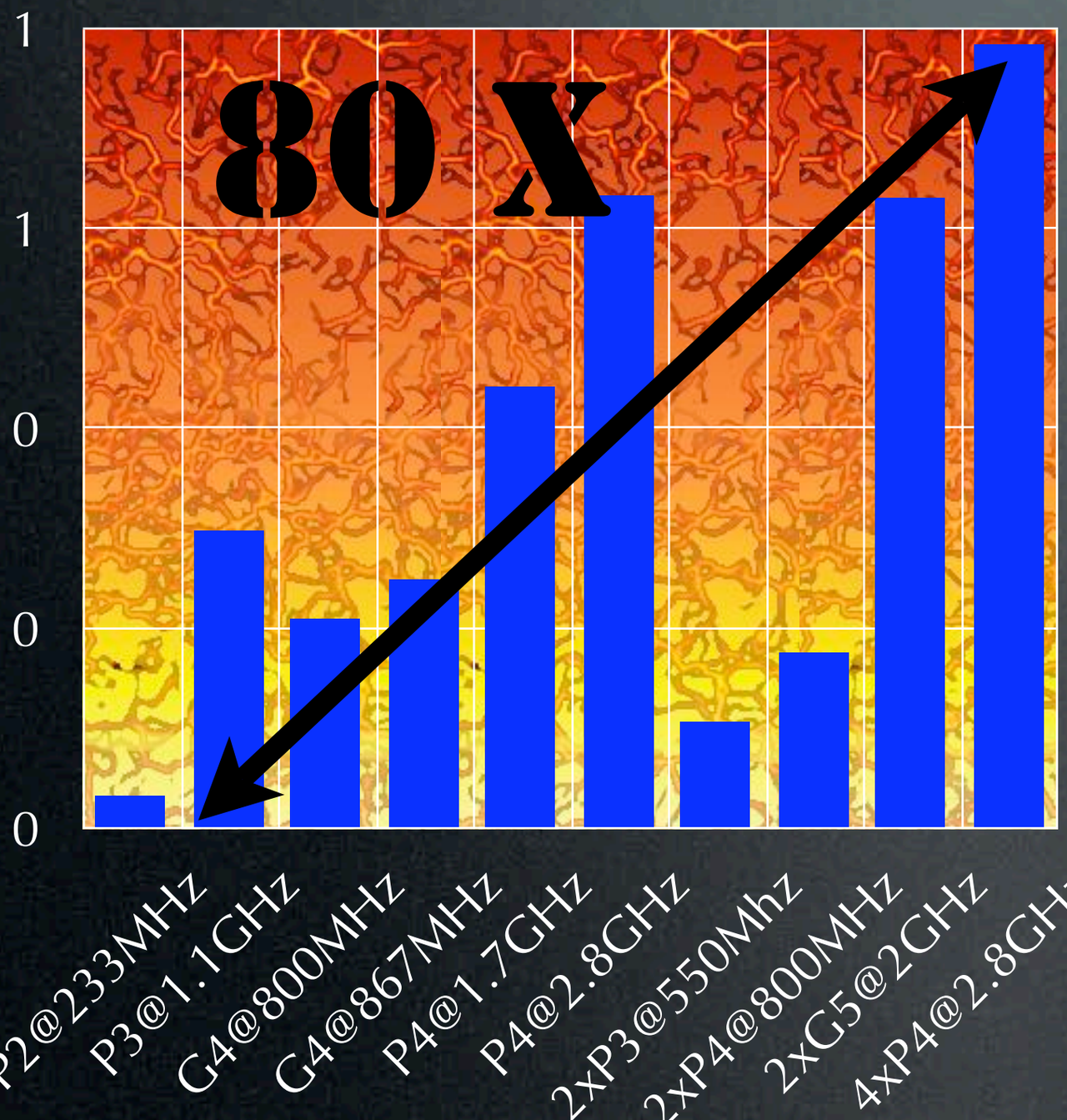
ASSIST in 1 minute



green boxes are processes

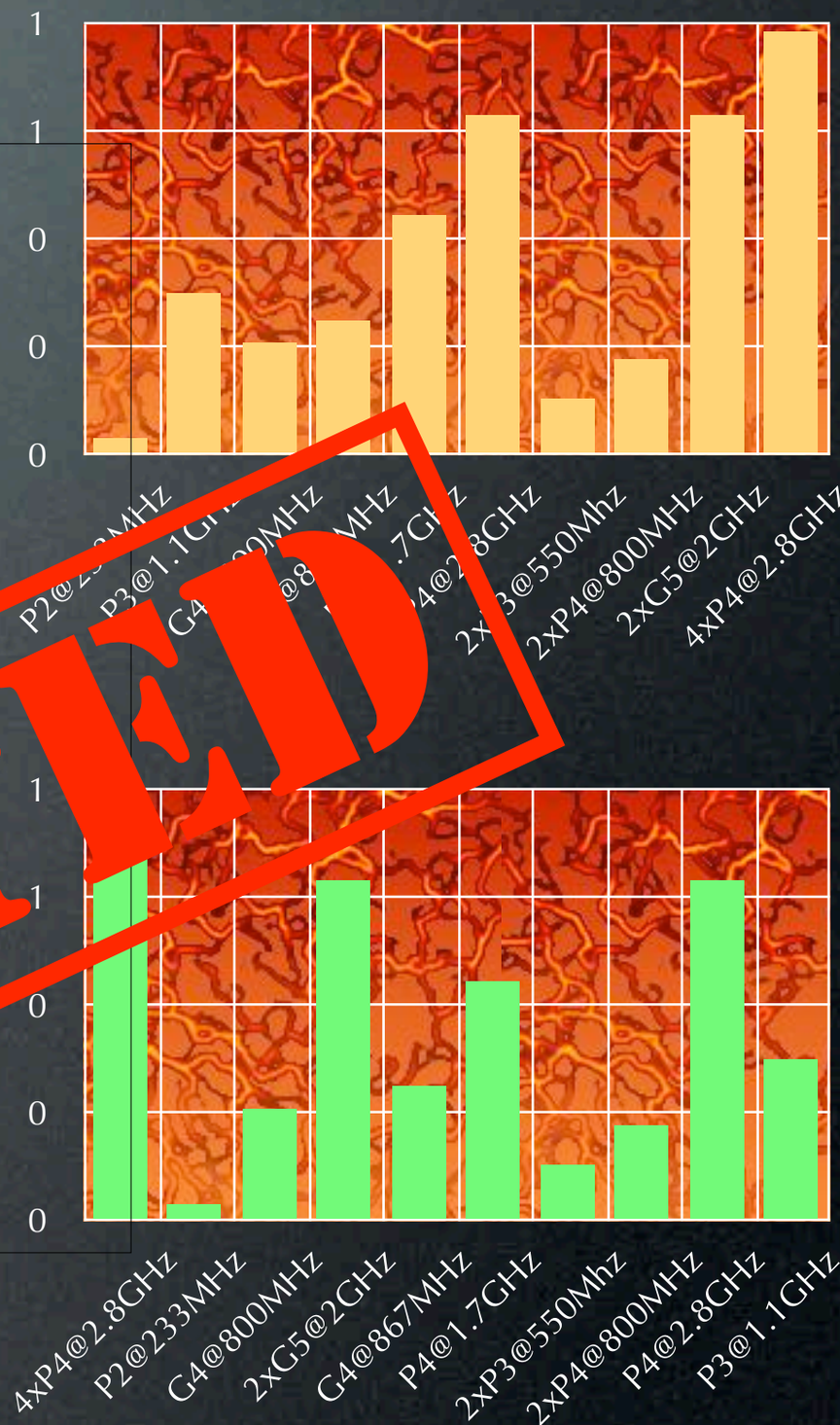
Motivating adaptivity

Boxes performances



- Grid platforms are supposed to exploit different “power” (in the meaning of Aristotelic power/act)
- and net bandwidth
- both of them may rapidly change over time

Performance metrics



REJECTED

Speedup ... ?


10
8
6
4
2
0
Ideal
1 2 5 6 8 9 10
45
36
27
18
9
0
Ideal
1 2 3 4 5 6 7 8 9 10
REJECTED

QoS Contract
(where CS meets c-business)

- Logical formula with constraints
  - Service time, PEs number and kind, ...
- Entertainment in UK
  - the software should elaborate on-demand of 5 OLAP entries per second per 100 users for the 98% of running time
  - the software should be seamlessly portable on clusters, NOW, single PE, SMP to meet different customers needs

Aldinucci, Gusmeroli, Vanneschi, Villa
[ACM INTEROP-05 (accepted) , FMOODS/DAIS-05 (submitted), e-challenges-05 (submitted)]

Adaptivity

Program adaptivity

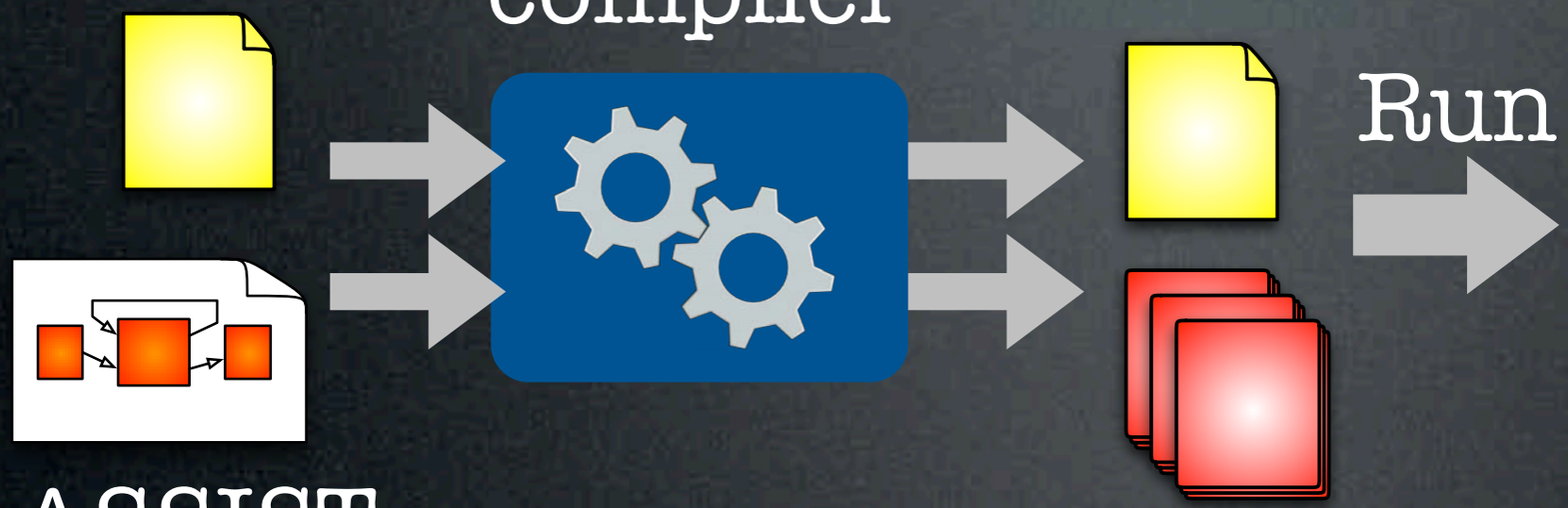
- Adaptivity aims to control program configuration (e.g. parallel degree) and mapping
  - for performance (high-performance is a natural sub-target)
  - for fault-tolerance (enable to cope with unsteadiness of resources, and some kind of faults)

Executing programs

QoS contract

ASSIST compiler

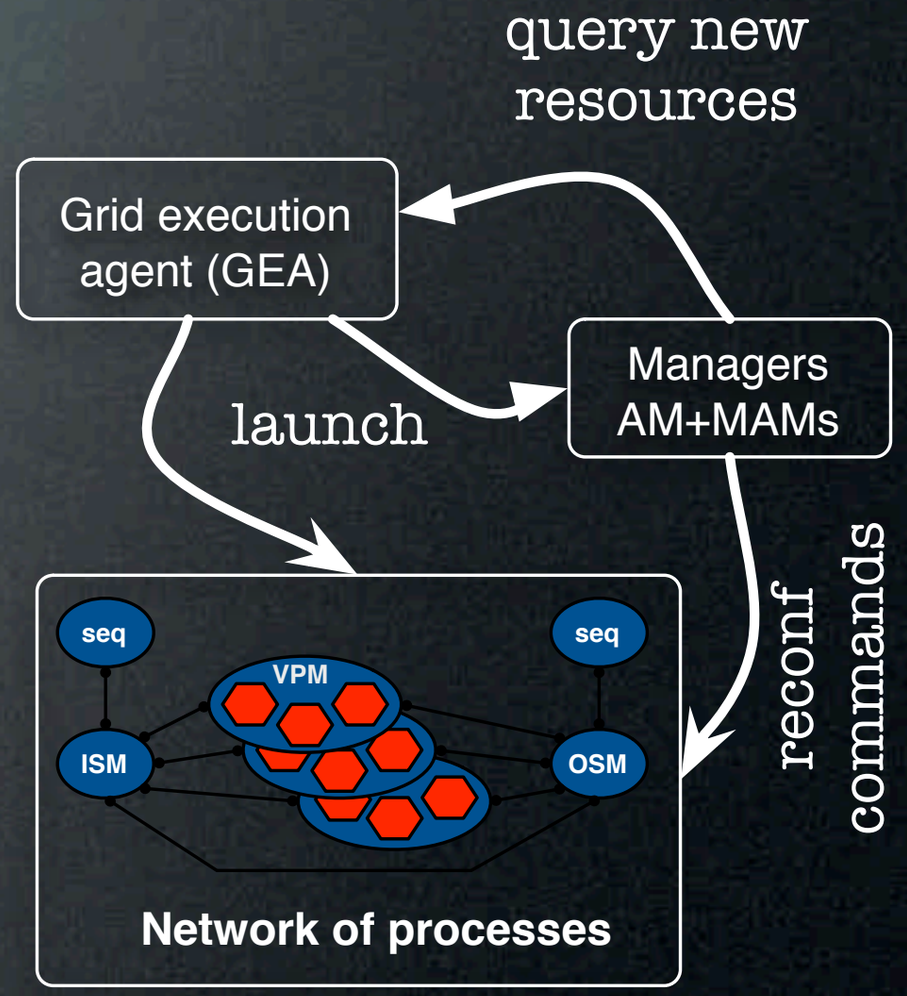
resource description XML



ASSIST program

executable code (linux, mac, uindoz)

Run



Run-time computation re-shaping

1. Mechanism for adaptivity
   - reconf-safe points
     - where place them? Who place them?
   - reconf-safe point distributed agreement
   - add/remove/migrate
     - what? (processes, threads, data,...)
2. Policies for adaptivity
   - QoS contracts
     - Describes QoS qualified data for components/applications
   - "self-optimizing" components/module
     - under the control of managers, which are hiearchically organized (Application Manager being the root)
     - Different levels manage different aspects of QoS control

Mechanisms

reconf-safe points

- In which points of the code the execution can be reconfigured?
  - Parallel entities exploit an intrinsic coherence
- 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

reconf-safe points /2

- transparent to the programmer
- defined to match “natural” synchronization points of the parmod
  - on-stream-item
  - on-barrier
- no artifactual synchronization added
  - already existing synchronizations are rather instrumented
  - overhead w.r.t. not adaptive code < 0.04%

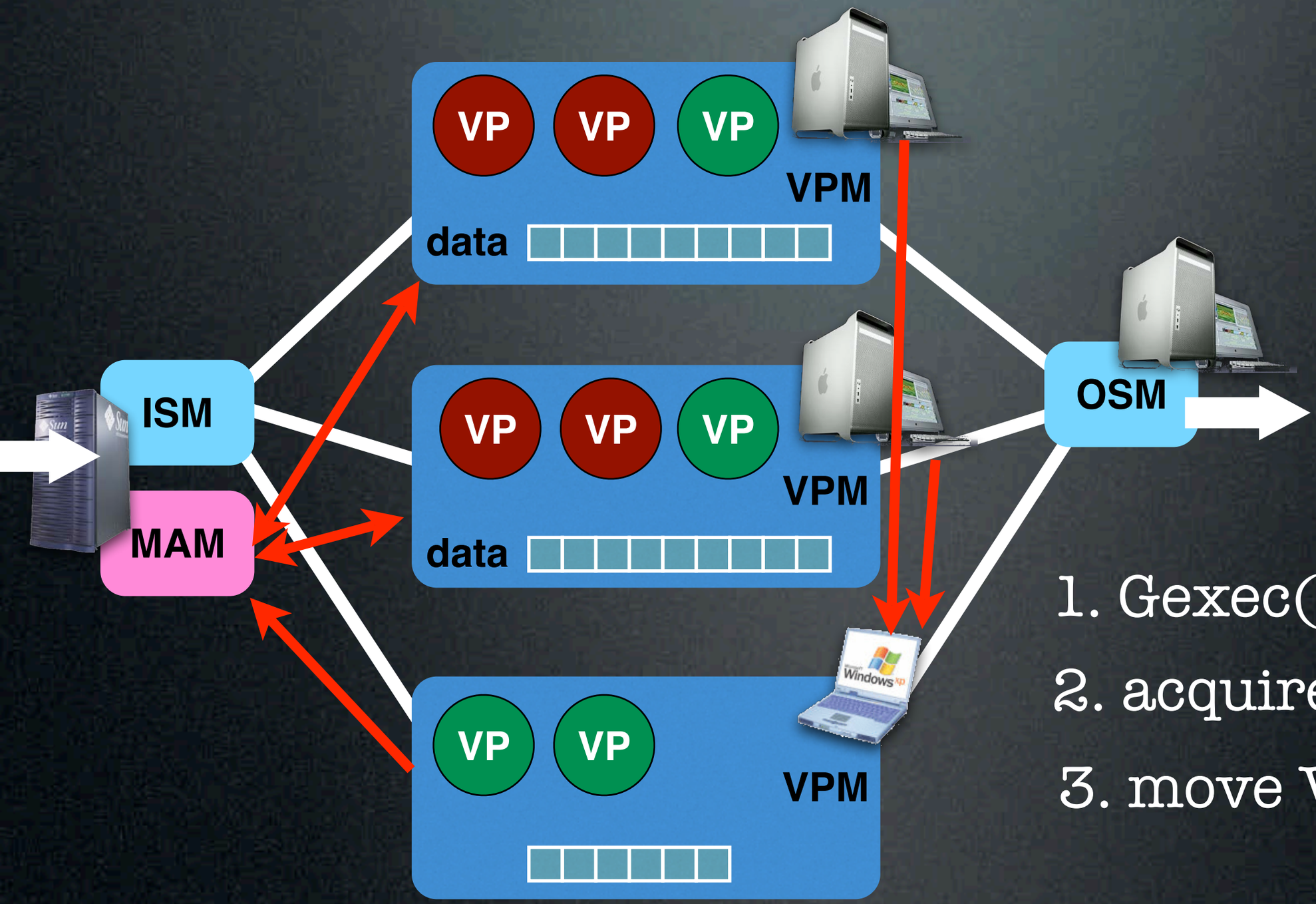
Distributed agreement

- 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
  - possbily 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
  - Adaptive Load-balancing as sequence of Move operations

Example: Add VPM



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

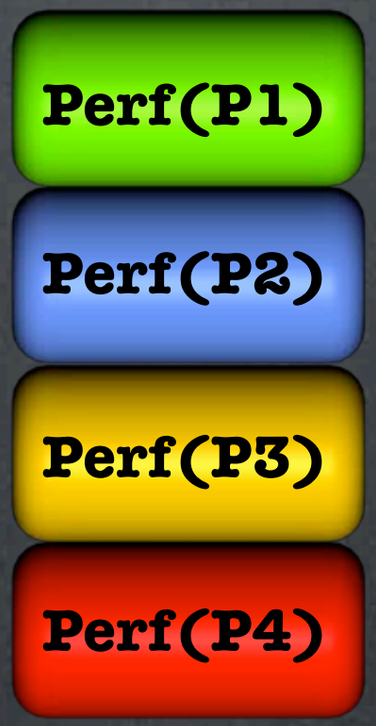
Only 3. is in the critical path

Overheads of mechanisms (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 hundres of seconds (K. Kennedy et al. 2004)

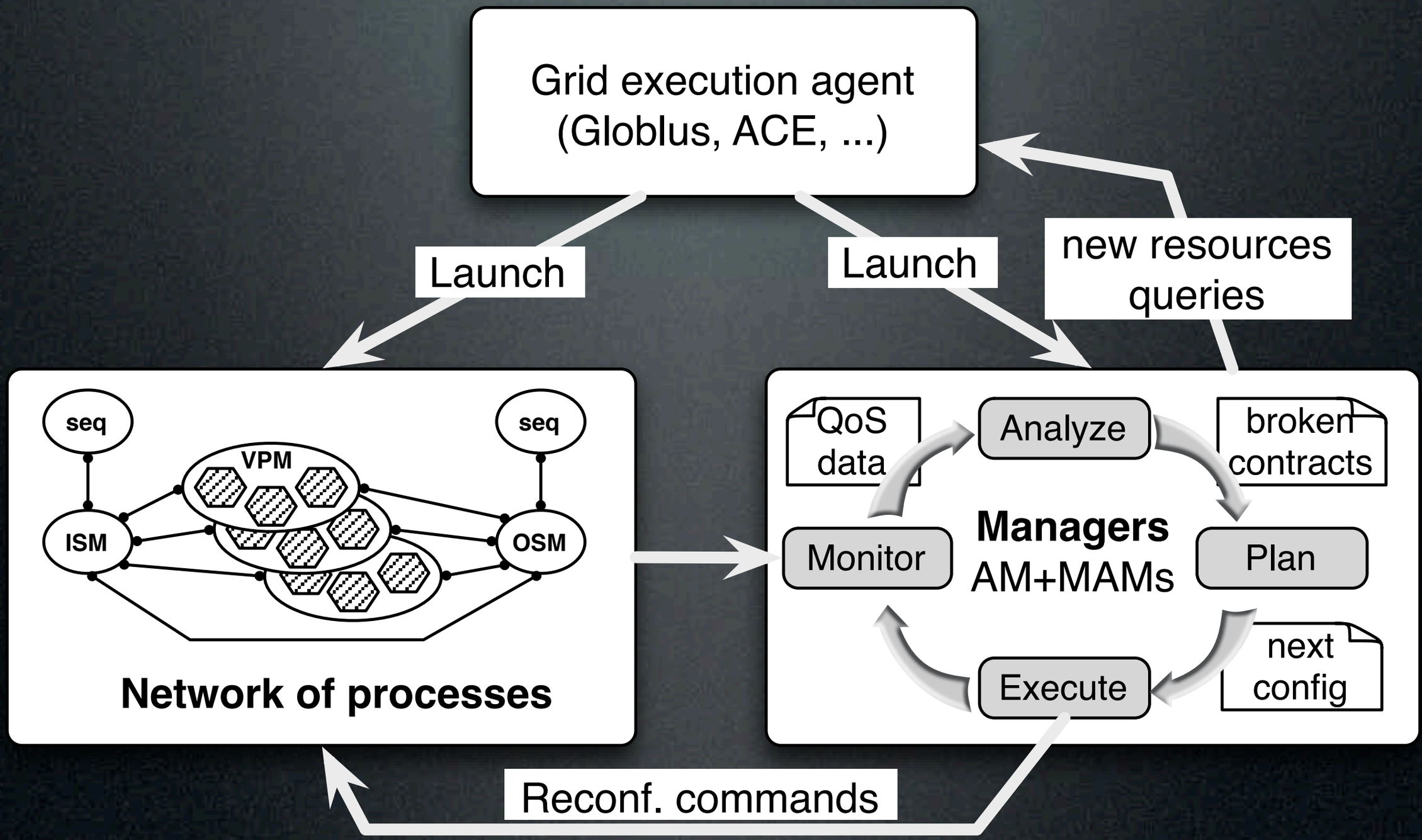
Management, Perf. Models & Policies



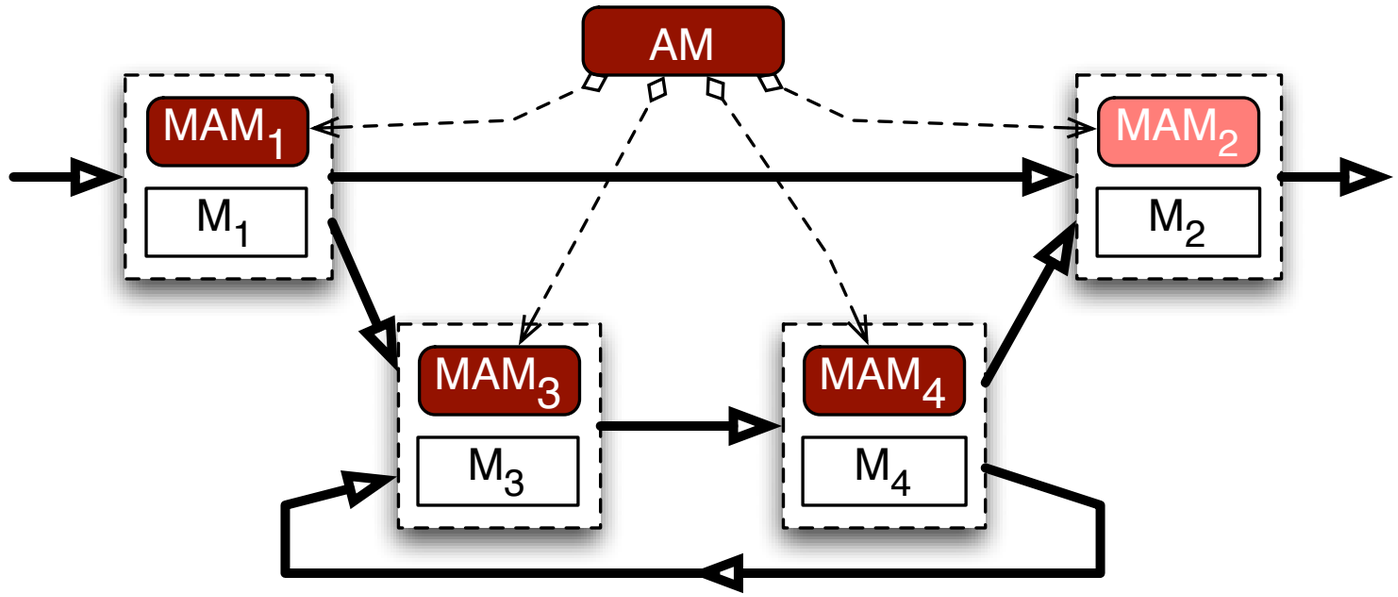
High-level applications

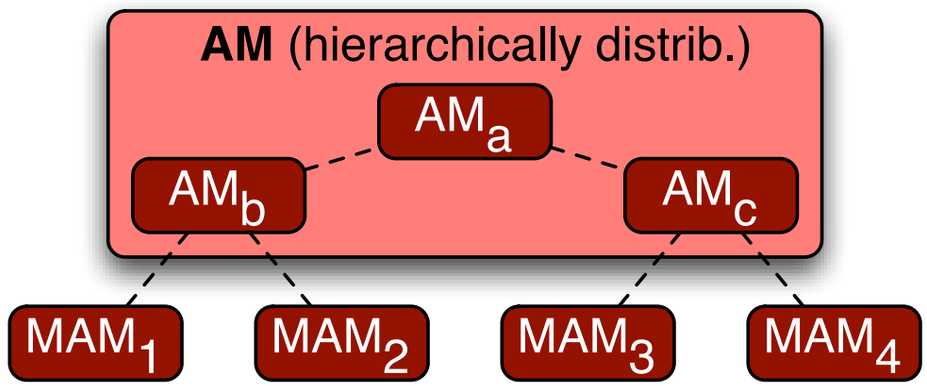
- Policies are developed starting from computational paradigms
- Mechanism are highly optimized taking in account program semantics
- because it is high-level, and structured
  - even if there is no deal, pipe, farm, haloswap, etc keywords
- it is a cultural heritage of skeletons community

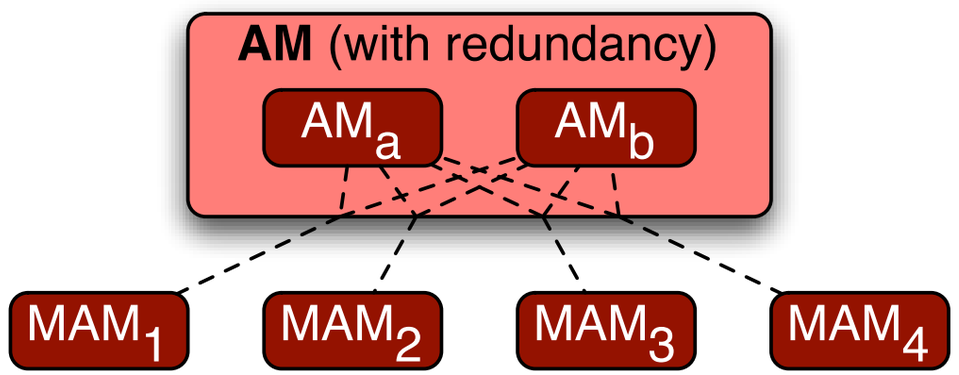
Autonomic Managers



Hierarchy of managers



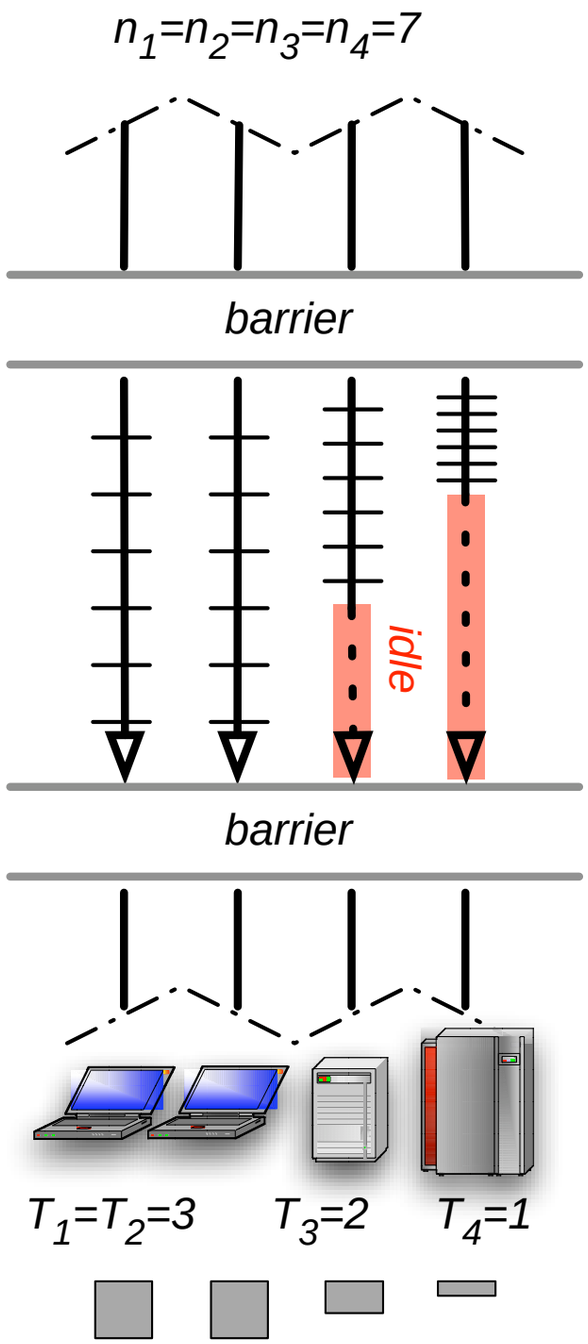


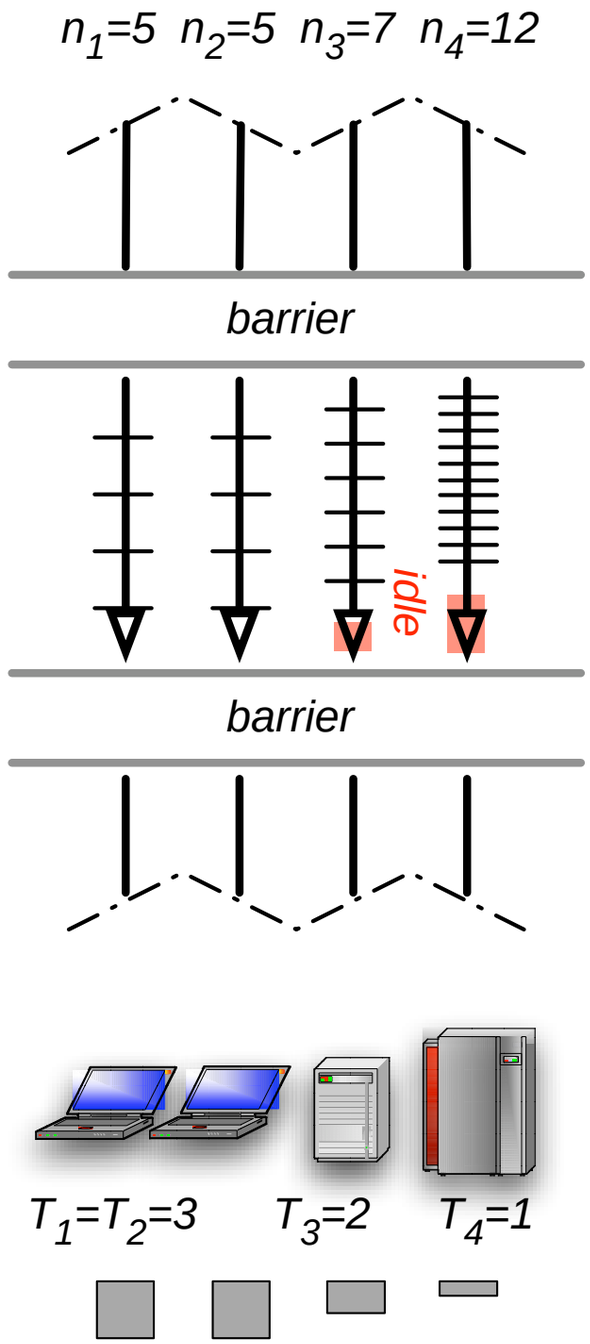


Why hyerarchical

- Enanche locality of decisions
  - match Grid cluster-of-clusters structure
  - avoid single point of failure
- Distinguish responsibility of tiers
  - tier-0 provide load-balancing within the single parmod
  - upper tiers provide higher-level policies (e.g. rebalancing across parmods)
- Deal with loops
  - insulate subgraphs that needs stationary configurations
  - Anne, Jane: Help! (blackboard)

Performance models: an example (DP load balancing)



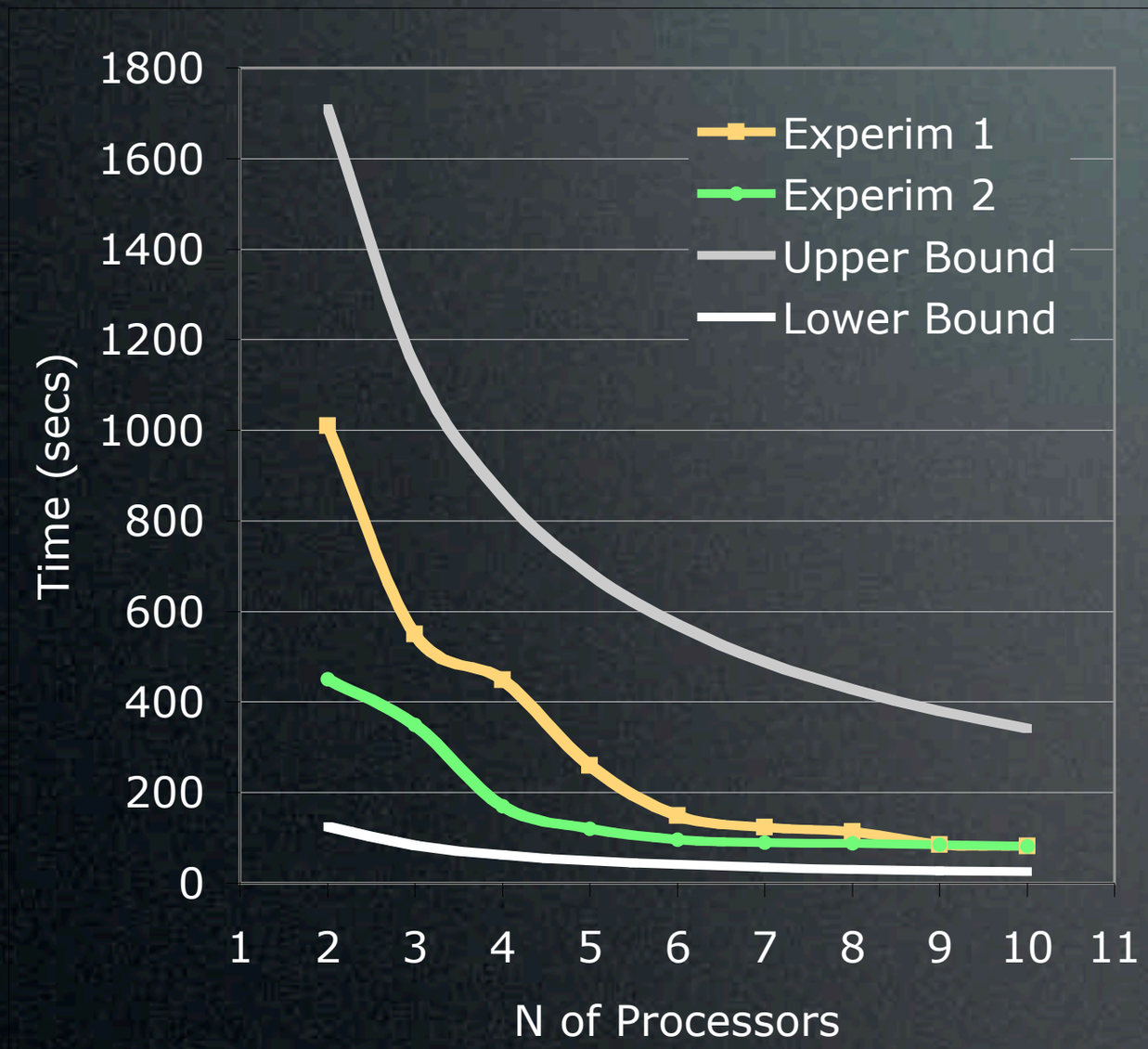


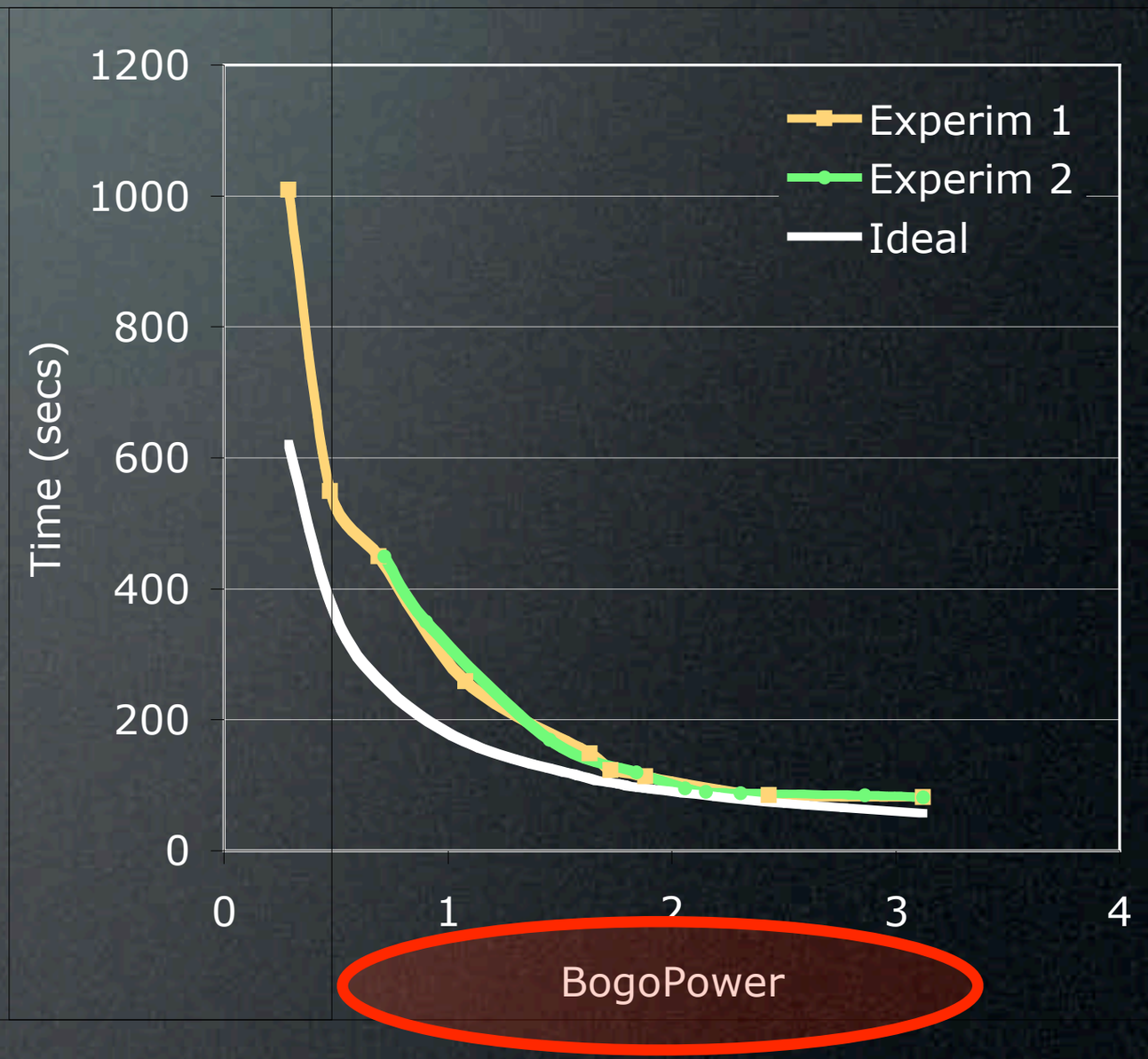
A simple formalization

$$\begin{cases} \sum_{i=0}^{m-1} n_i = n \\ \forall i,\ n_i t_i = t \end{cases} \Rightarrow \begin{cases} H_t = \dfrac{m}{\sum_{i=0}^{m-1} \frac{1}{t_i}} \\ n_i = \dfrac{n H_t}{m t_i} \end{cases}$$

n_i	num. of VPs mapped onto VPM_i
m	num. of VPMs
t_i	execution time of VPM_i
t	execution time for the next RWindow
n	total num. of VPs
H_t	harmonic mean $H(t_0, \cdots, t_{m-1})$

Two experiments revised





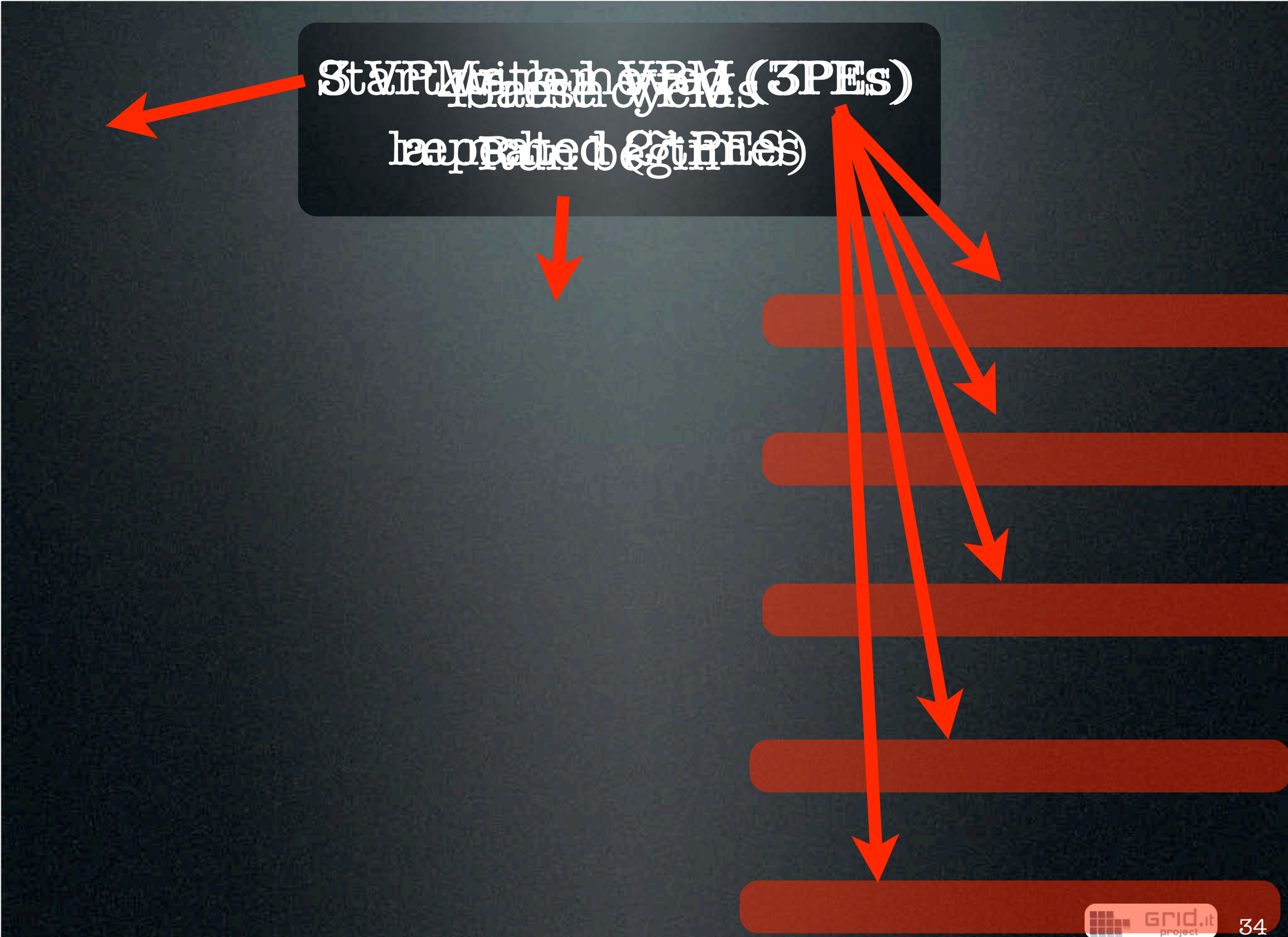
Some experiments

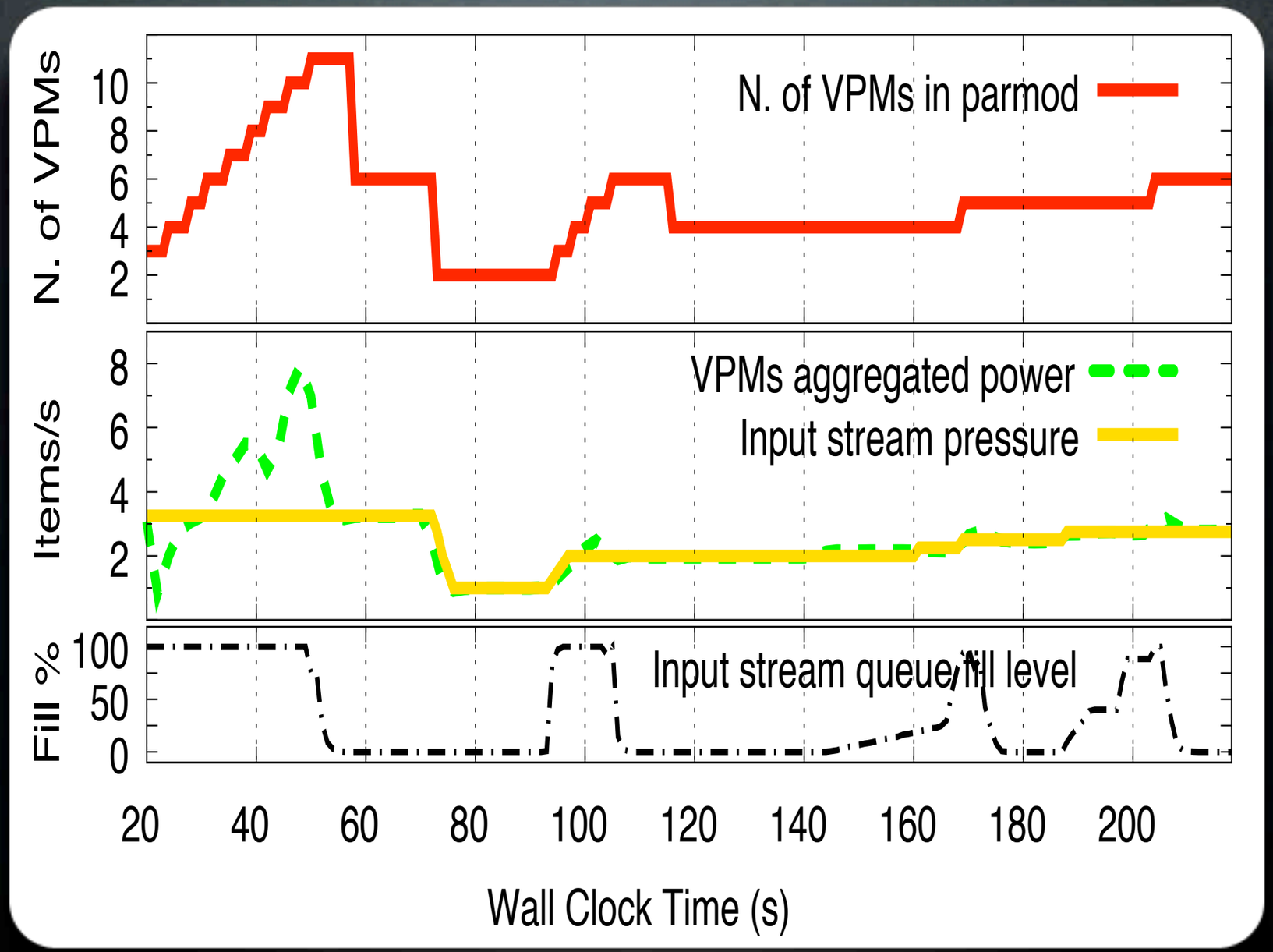
Start VM with 3 VCPUs

Start VM with 1 VCPU

Start VM with 2 VCPUs

Start VM with 3 VCPUs

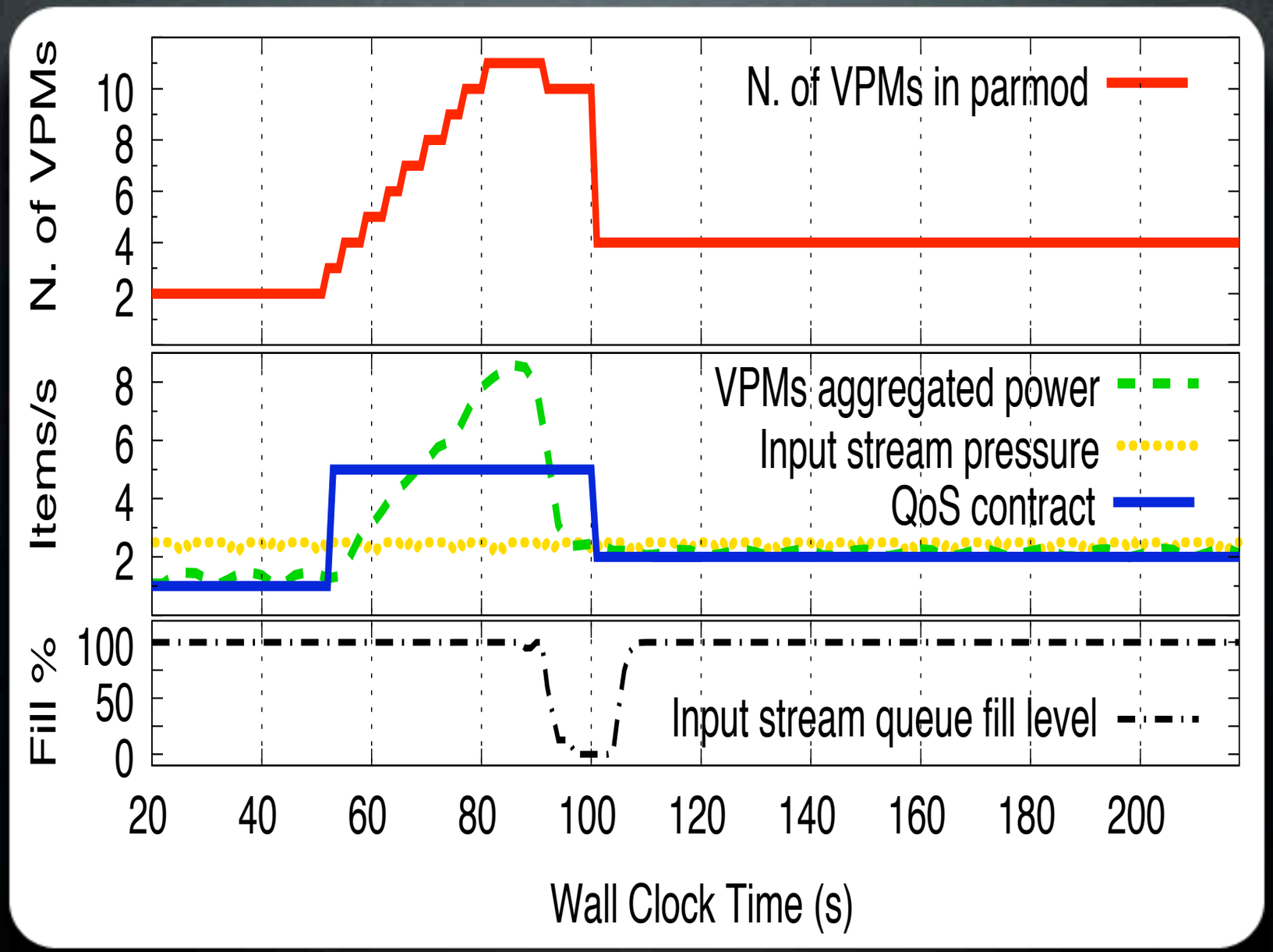




Farm:

contract: optimize time-resources tradeoff

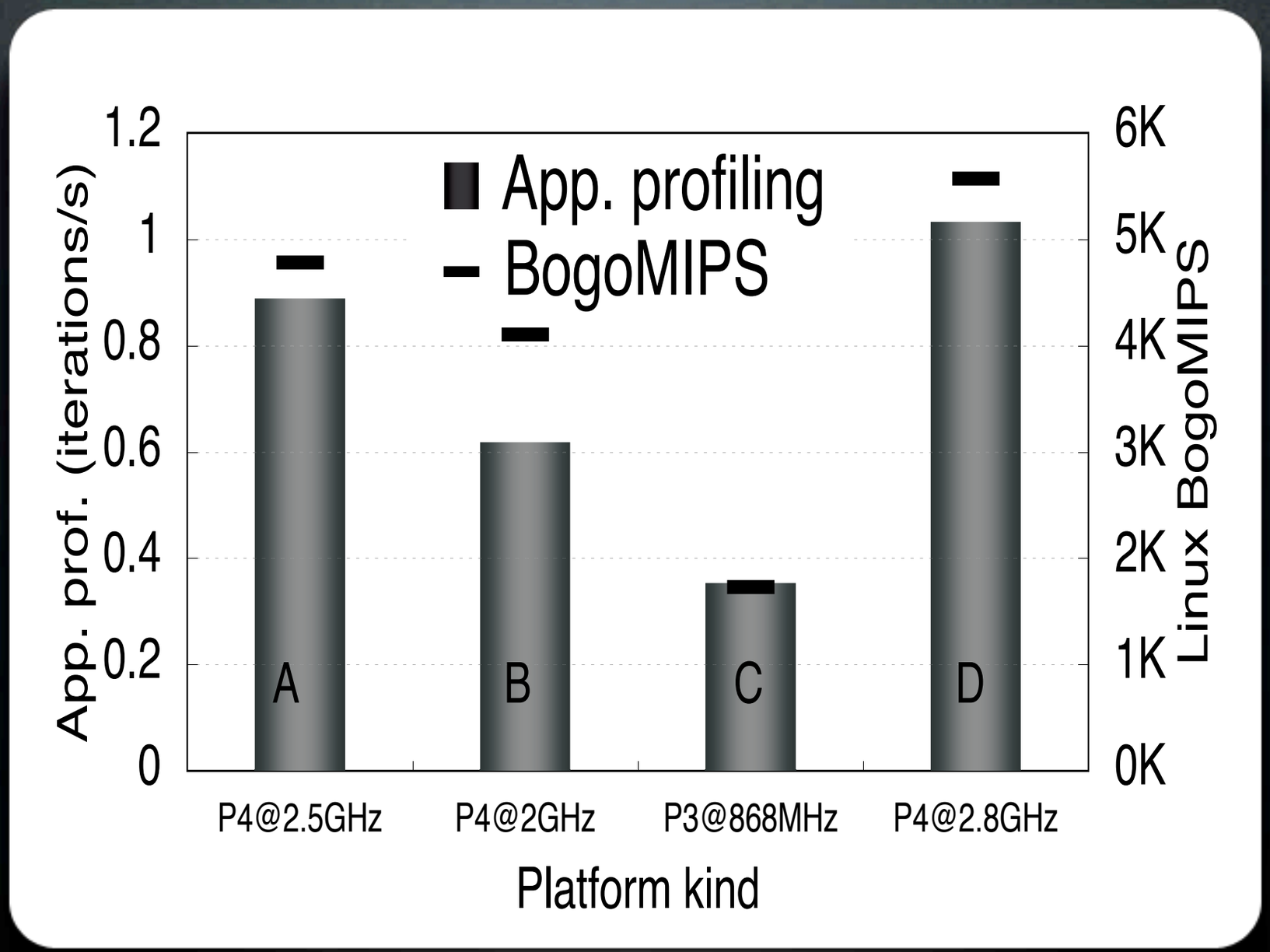
byproduct: reduce load spikes



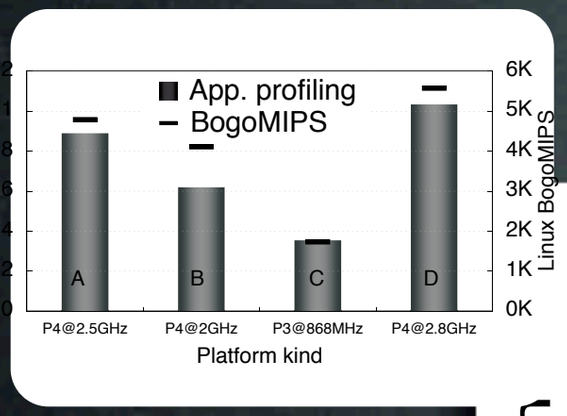
Farm:

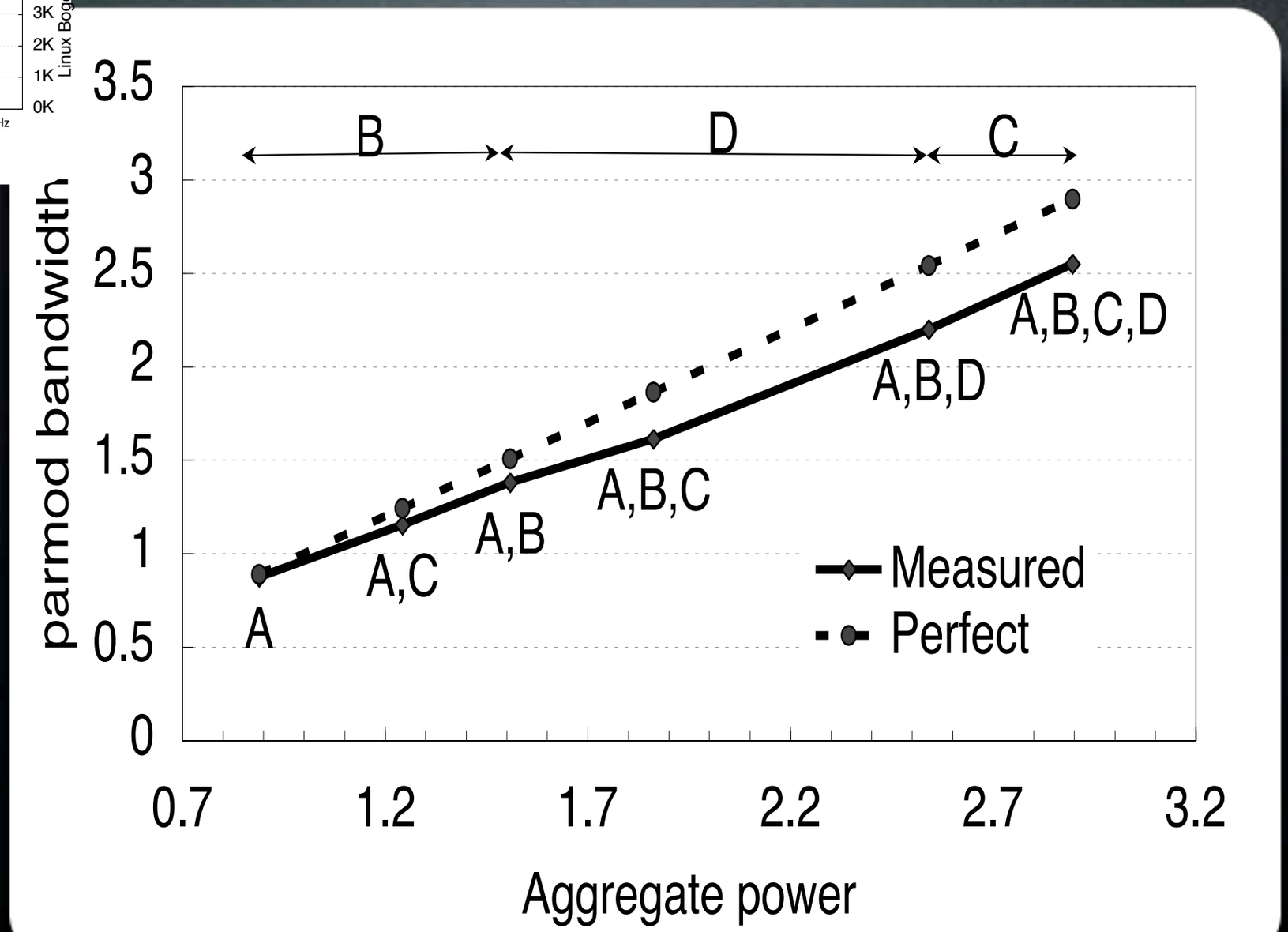
contract: keep a given service time

contract change along the run

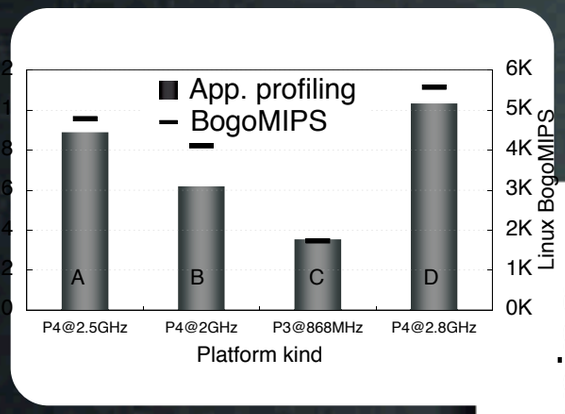

App. profiling
BogoMIPS
App. prof. (iterations/s)
Linux BogoMIPS
1.2
1
0.8
0.6
0.4
0.2
0
6K
5K
4K
3K
2K
1K
0K
A
B
C
D
P4@2.5GHz
P4@2GHz
P3@868MHz
P4@2.8GHz
Platform kind

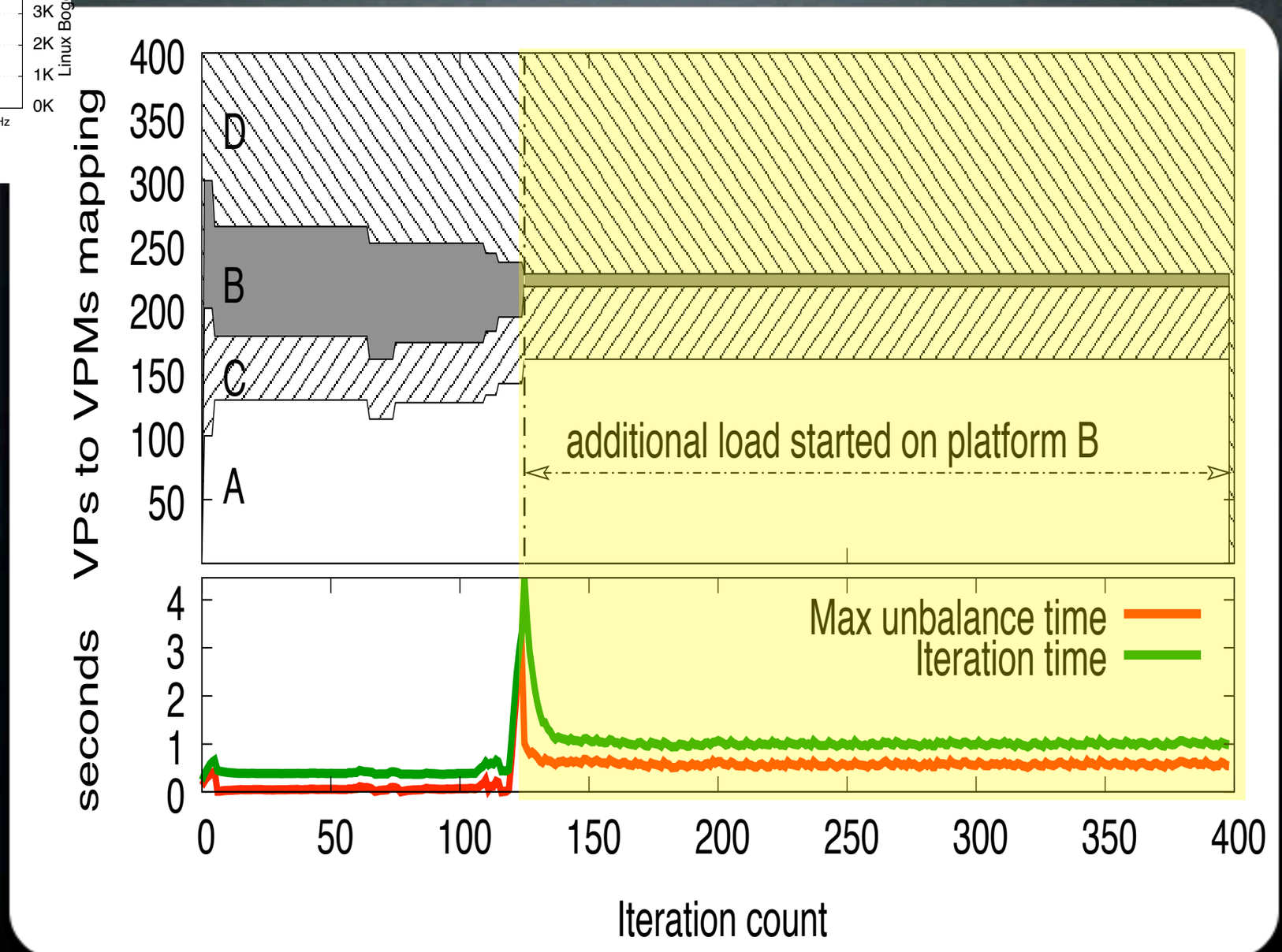
Running Env



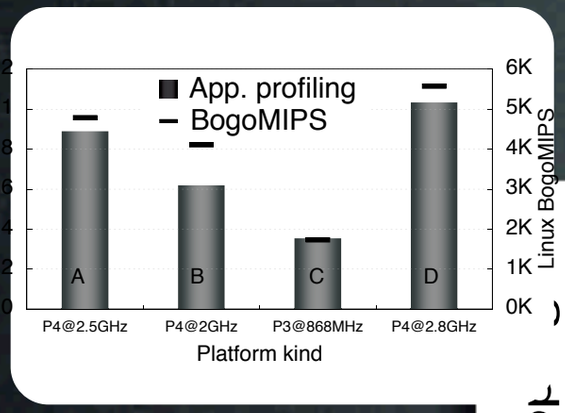


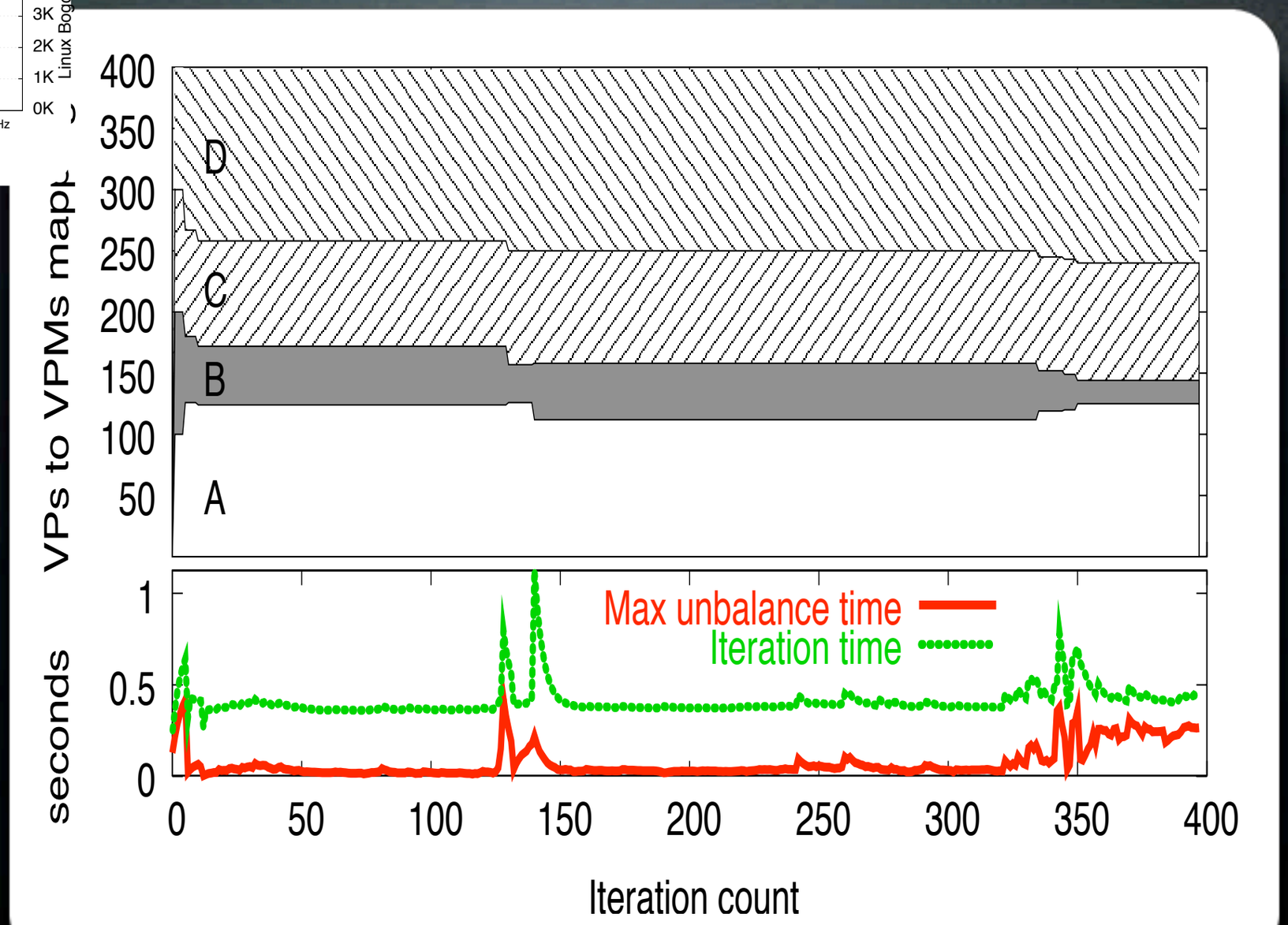
Forecast short-term performance
(and evaluate it against QoS contract)





Data parallel (shortest path)
Machine externally overloaded





Data parallel shortest path workstations in typical daily usage

References

- ASSIST started in 2001
  - developed across several projects
  - http://www.di.unipi.it/groups/architetture/
- Adaptivity started in 4Q-2004
  - very ongoing work
    - almost all publications are currently submitted (Aldinucci et al.: Europar-05, FGG-05, INTEROP-05, FMOODS/DAIS-05, ParCo-05, e-challenges-05, ...)
    - mail me if interested: aldinuc@di.unipi.it
  - however
    - mechanisms already at stable release (v 1.3) on Linux/Mac, and heterogenous clusters of them
    - policies alpha testing stage
    - supports for TCP and Globus 3.0

ASSIST is the result of the collective effort
of several persons, I owe thanks to them, and

to you