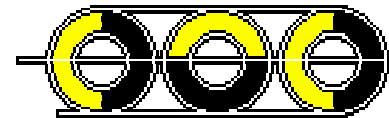


Grid-based Performance Control for Scientific Coupled Models

Chris Armstrong, Rupert Ford, Len Freeman,
John Gurd, Mikel Luján, Ken Mayes, Graham Riley

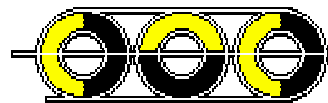
APART Workshop SC-2004



Centre for Novel Computing
www.cs.man.ac.uk/cnc

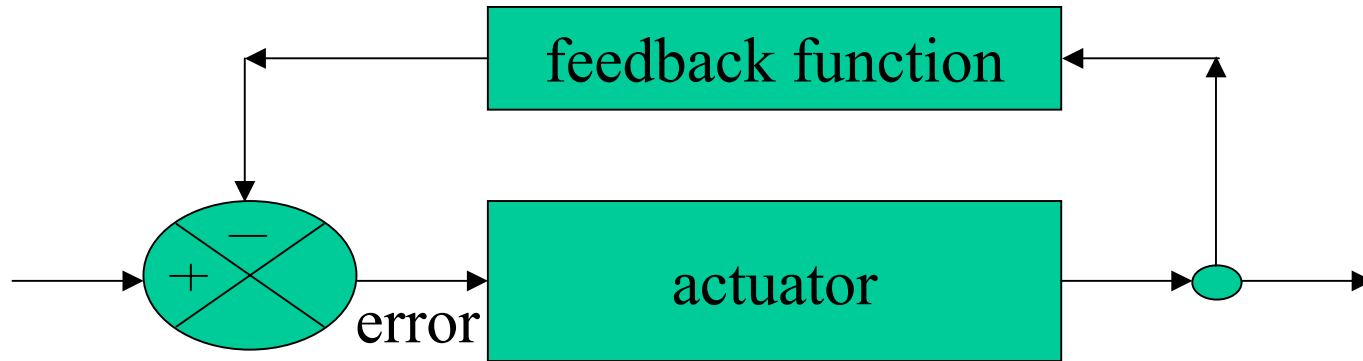
The case for Performance Control

- Performance Analysis
 - Objective: search for maximum performance
 - coarse design, then fine tuning
 - requires high degree of repeatability
 - benefits from homogeneity, symmetry, etc.
- Performance Control
 - Objective: control to achieve (less than max.) target
 - use negative feedback control at run-time
 - necessary to approach dynamic environment
 - helps to deal with heterogeneity

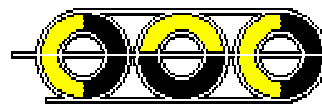


How to Control Performance?

- Requires (negative) feedback

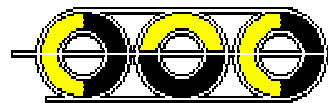


- needs sensors, actuators and compensators
- timers, control 'handles', predictive models
- Whole system vs. piece-wise control
 - who is responsible for what?
- Perception is that a hierarchy is needed
 - hence need hierarchical control structure



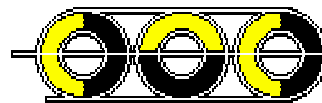
Overview

- Preamble
 - The case for Performance Control
- **Context**
 - **Malleable, component-based Grid applications**
- Components from Coupled Models
 - A simple example
- The PERCo (Performance Control) System
 - Design and status update
- Conclusion



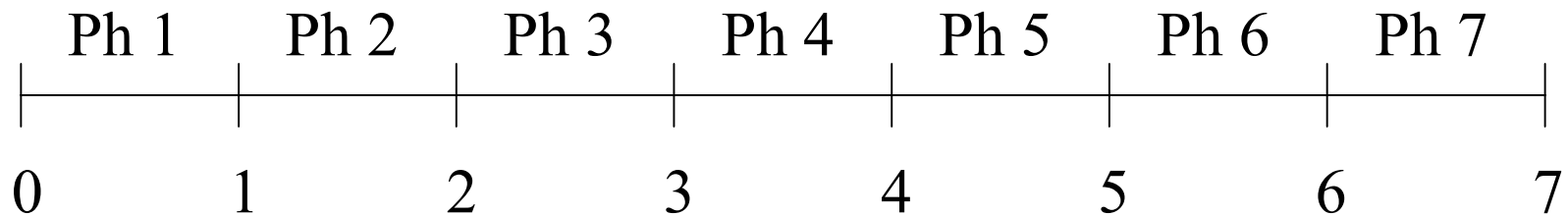
Controllable Components/Applications?

- Several groups have suggested component-based software for the construction of Grid applications
- *Malleable components* are components that provide control handles (operations) to affect their run-time performance
 - e.g. change number of processors, or block size.

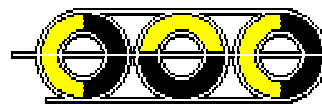


Progress Points

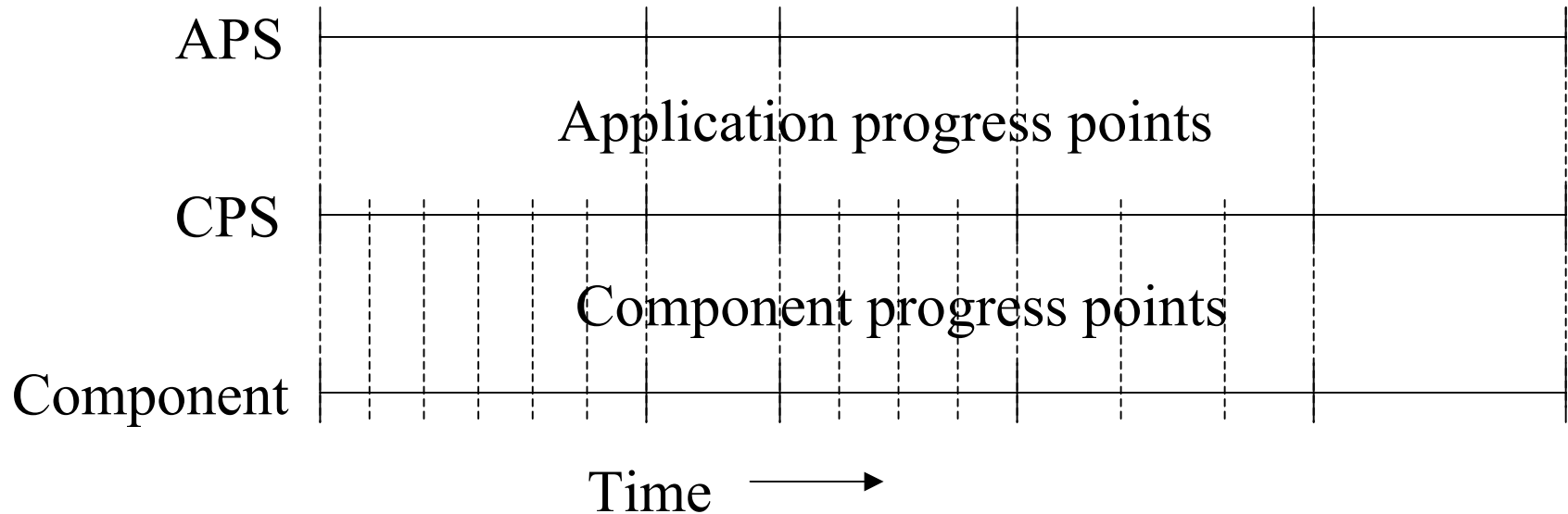
- Assume that the execution of components and application proceeds through cyclical *phases*, and that the phase boundaries are marked by *progress points*



- Can take decisions about performance and (possibly) actuate at the progress points



Two Levels of Progress Points

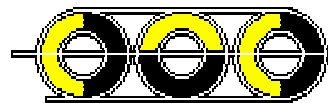


- Leads to *minor cycles* and *major cycles*
- Application progress points also need to be *safe points* (components can be redeployed)



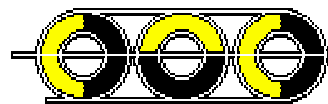
Controllable Components/Applications?

- Several groups have suggested component-based software for the construction of Grid applications
- *Malleable components* are components that provide control handles (operations) to affect their run-time performance
 - e.g. change number of processors, or block size.
- But where do the components come from?
 - a knotty problem (cf. RealityGrid LB3D)



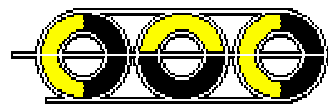
Overview

- Preamble
 - The case for Performance Control
- Context
 - Malleable, component-based Grid applications
- **Components from Coupled Models**
 - A simple example
- The PERCO (Performance Control) System
 - Design and status update
- Conclusion



Coupled Models

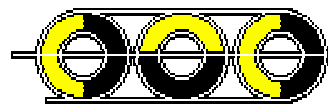
- Many scientific modellers are finding a need to link together multiple models:
 - climate/envt. models (ocean + atmosphere + ...)
 - aircraft lightning strike (CEM + a/f structure)
 - multi-scale phenomena (CFD + MD = HybridMD)
 - + others, all needing high performance & 'Grid'
- The individual models seem to constitute ready-made components.



Coupled Models

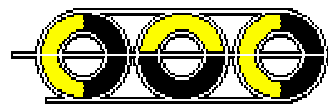
- Many scientific modellers are finding a need to link together multiple models:
 - climate/envt. models (ocean + atmosphere + ...)
 - aircraft lightning strike (CEM + a/f structure)
 - multi-scale phenomena (CFD + MD = HybridMD)
 - + others, all needing high performance & 'Grid'
- The individual models seem to constitute ready-made components.

Can these models be used for performance control?



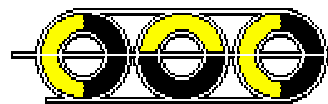
Generalised Coupling Framework (GCF)

- For maximum flexibility, GCF separates the coupled modelling task into three concerns:
 - **Describe** the individual models, which may be:
 - scientific models, or
 - transformer models
 - **Compose** different couplings & components
 - **Deploy** the components on various platforms
- XML metadata used at each level



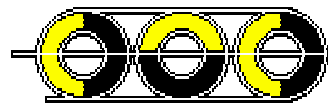
Bespoke Framework Generator (BFG)

- Processes the *GCF* metadata to generate code that implements the execution and communication mechanisms required by individual models in the coupled model
 - in prototype BFG, models are zero-argument subroutines or methods which compute a single time-step advance of the underlying science
 - coupling interface between models is via **get()** and **put()** calls



GCF Model Types

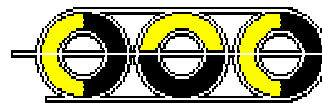
- Individual models step:
 - *1-for-1* scientific or (simple) transformer
 - { all get()s; work(); all put()s }
 - *n-for-1* transformer
 - { get()s; work() } every cycle, for n cycles, then put()s
 - *1-for-n* transformer
 - { get()s; work(); put()s } for first minor cycle (of n), then { work(); put()s } for remaining $(n-1)$ cycles
- Execution of associated **get()s** and **put()s** is *synchronised*



Connections and Fields

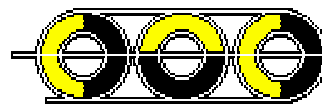
- Individual models are linked together by the *fields* that they specify in `put()` and `get()` commands
- Composition metadata defines *connections* between the fields of different models
- A *coupled problem*, CP , is a triple (M,C,F) where:
 - M is a set of individual models
 - C is a set of connections
 - F is a set of fields

Certain restrictions apply

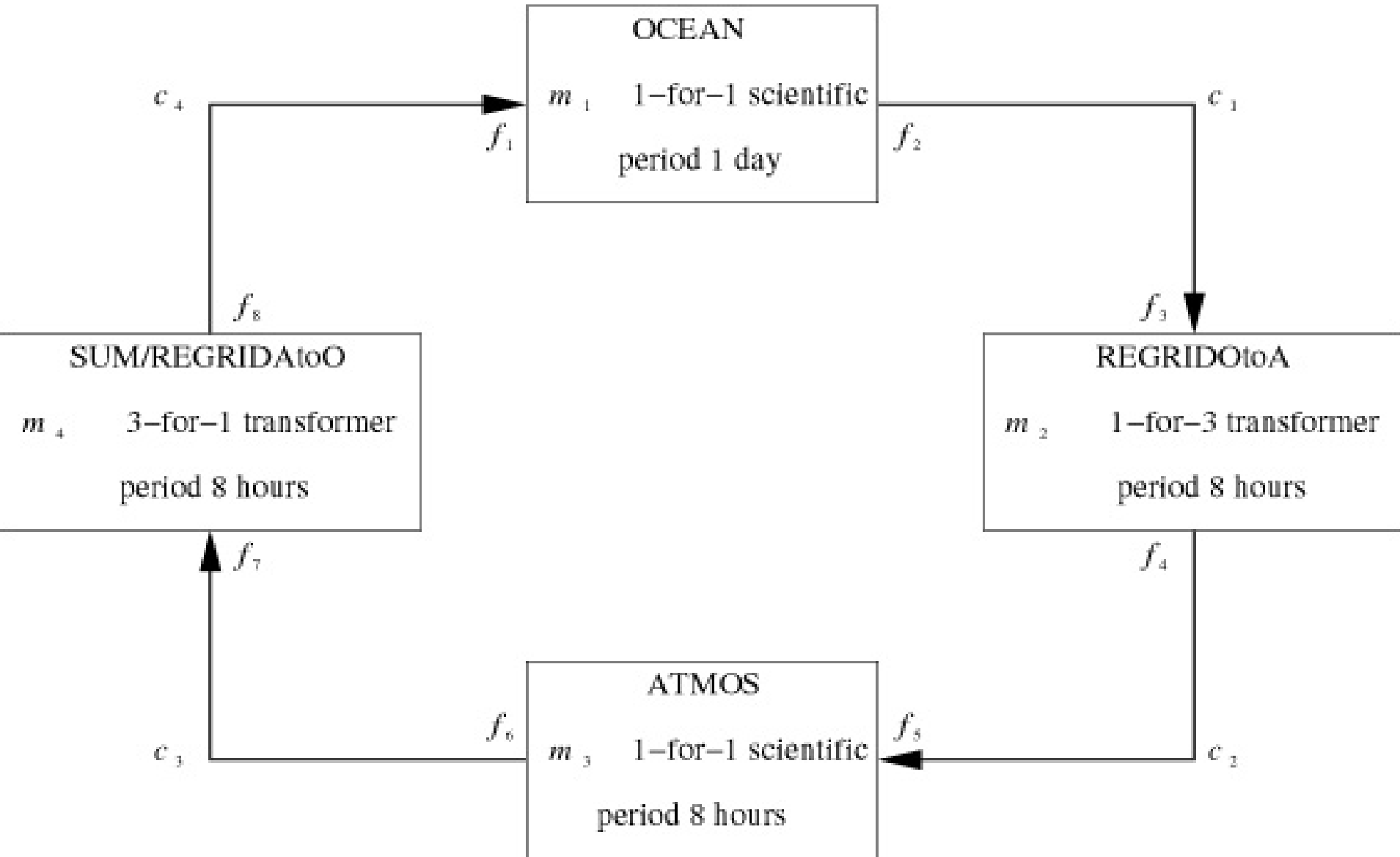


The Coupled Model

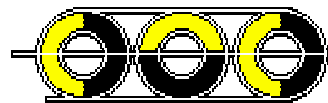
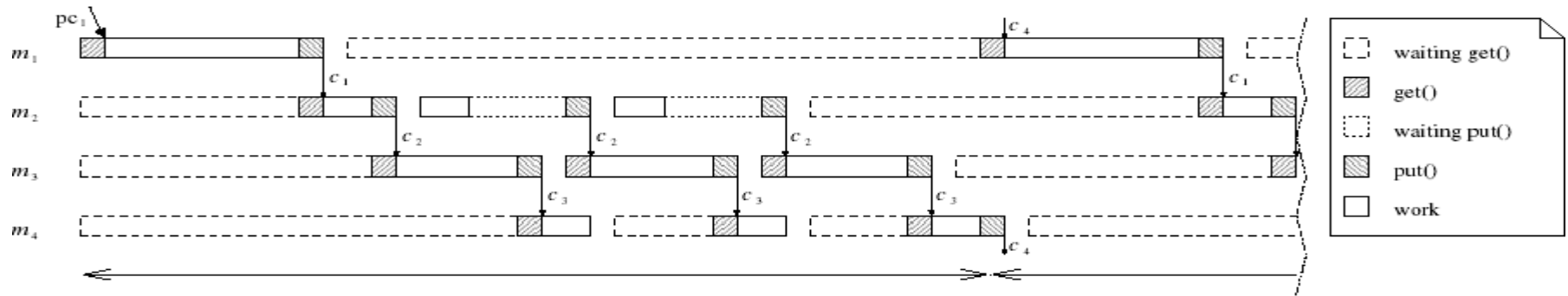
- A *coupled model* comprises:
 - its *coupled problem* (CP - see above)
 - its *duration* (in units of time)
 - its (set of) *initially primed connections*
- This is sufficient for the BFG to generate a code *harness* which will activate each individual model exactly when it is needed during each *major cycle* (of the entire coupled model), and then terminate when the specified duration is reached



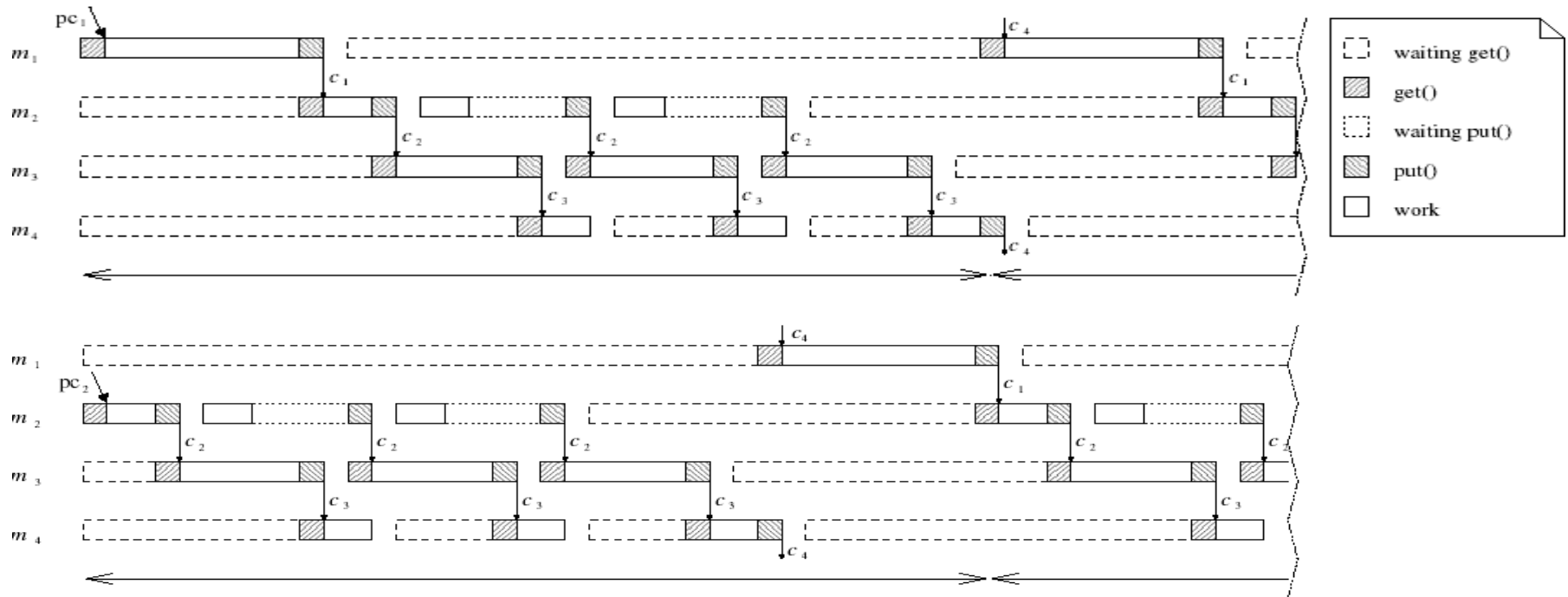
e.g. Ocean + Atmosphere



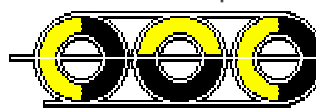
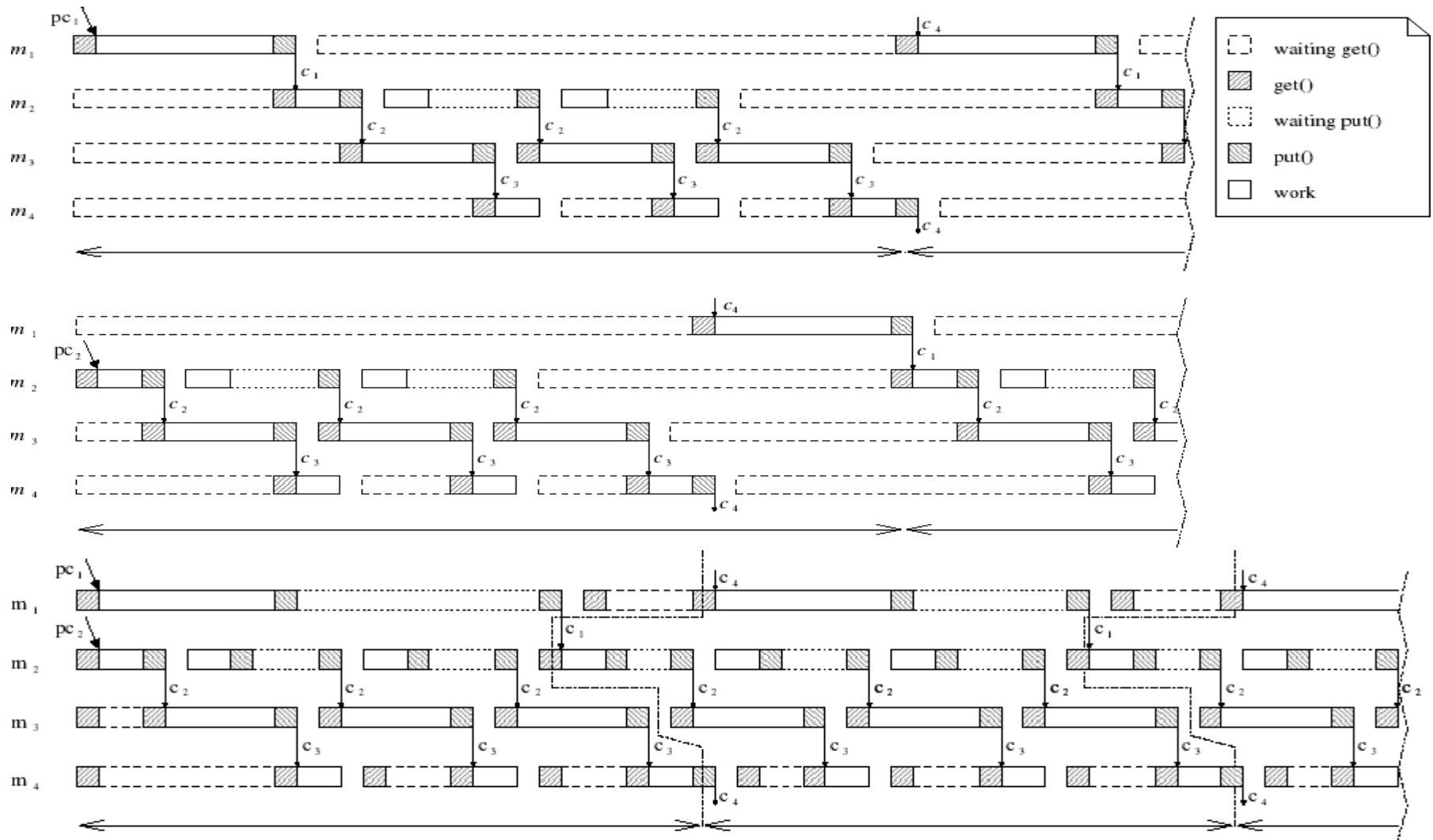
Possible Initial Primings



Possible Initial Primings

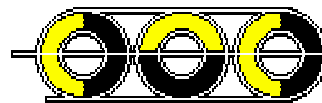


Possible Initial Primings



Overview

- Preamble
 - The case for Performance Control
- Context
 - Malleable, component-based Grid applications
- Components from Coupled Models
 - A simple example
- **The PERCo (Performance Control) System**
 - Design and status update
- Conclusion

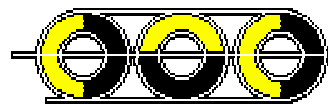
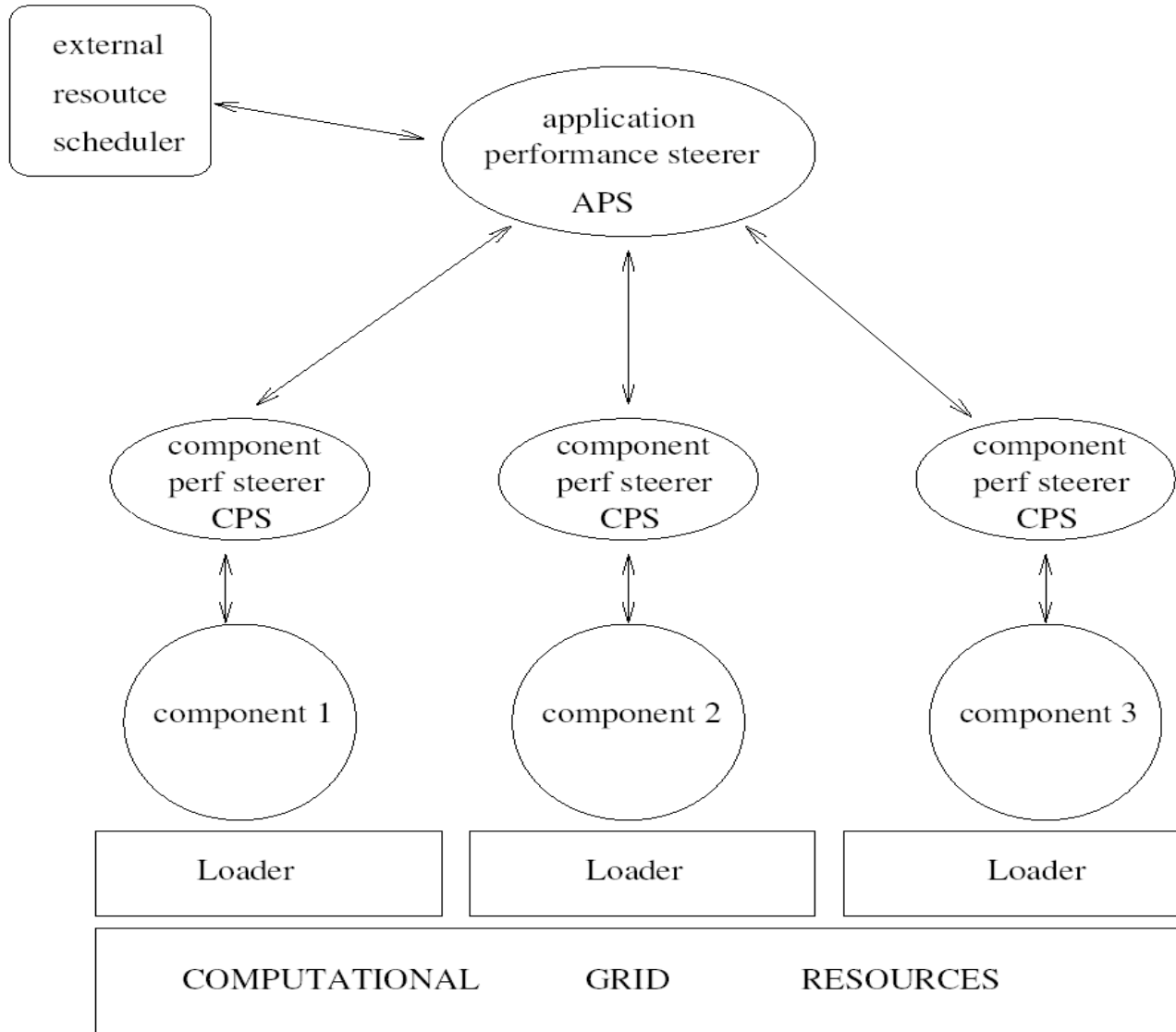


Overview of PERCO

- Two-tier hierarchical performance control
 - CPS (Component Performance Steerer)
 - one wrapped around each component
 - all attached to APS (see below)
 - maximises performance **on deployed platform**
 - APS (Application Performance Steerer)
 - (re)deploys components on available resources
 - maximises performance **on allocated platforms**
- Requires an external resource allocator

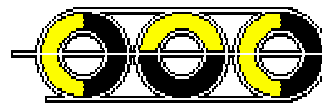


PERCO System Overview



Modus Operandi

- Components progress via a sequence of *progress points*, at each of which a component calls out to its CPS for any component-specific performance control actions (local actuation)
- Certain progress points are also *safe-points* (i.e. the component is in a state that permits it to be redeployed) and, at these points, the CPS can call out to the APS for redeployment-based performance control actions (the APS means of actuation)



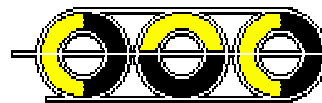
Coupled Models and PERCO

- In a *GCF* coupled model:
 - the components are the individual models
 - the end of each minor cycle is a progress point
 - the end of each major cycle can be a safe-point
- Thus a safe-point is defined to occur at the end of some fixed number of major cycles
- For the moment, no *CPS* action is taken at any progress point, so control is effected solely by means of redeployment via the *APS* at the safe-points (assume there are K of these)

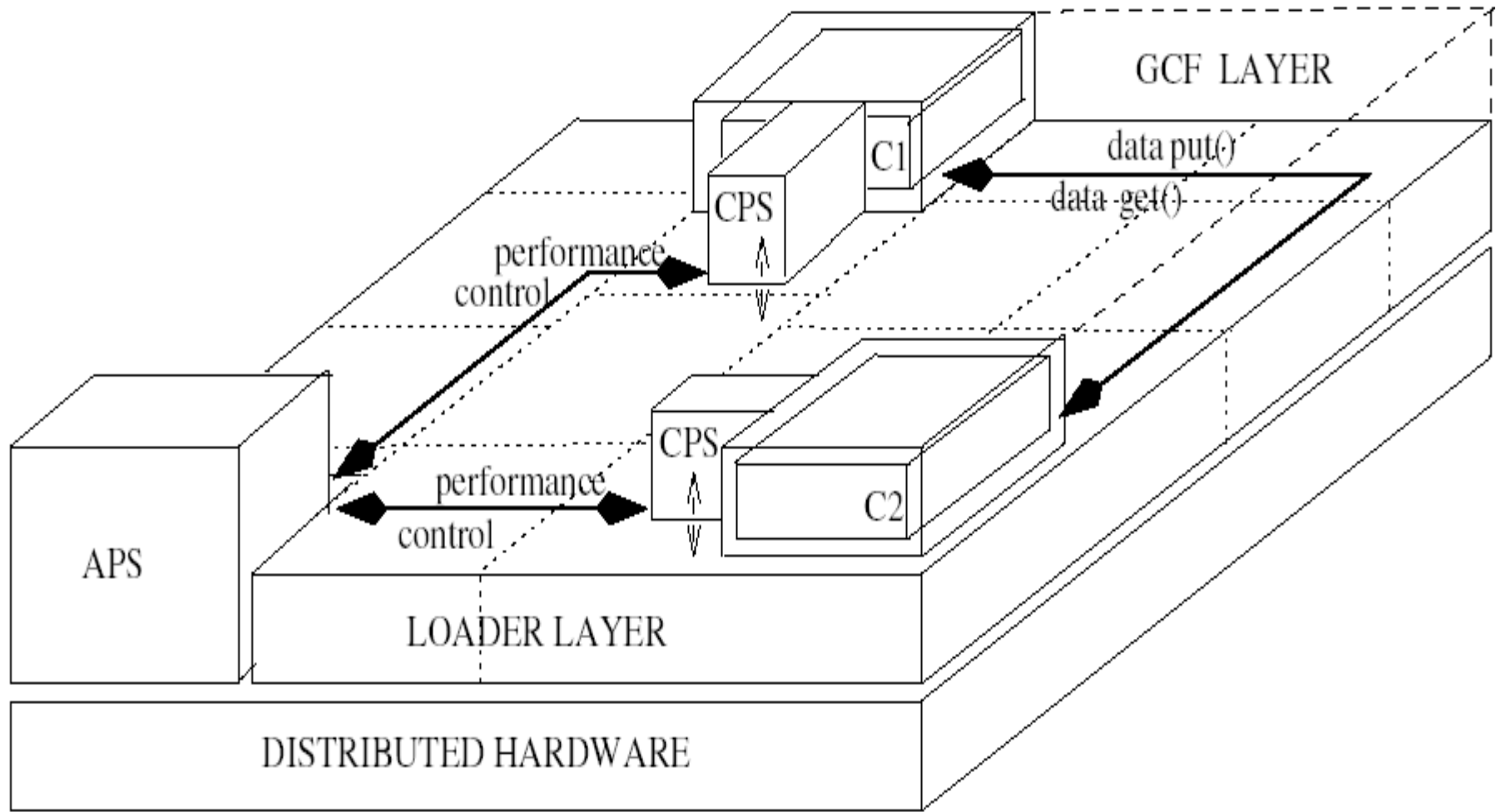


PERCO Infrastructure

- Each component is attached to a *local loader* which is capable of moving the component safely around the distributed Grid hardware according to the APS commands
- The local loaders act in concert with the APS to form a virtual *loader layer* for the application
- Each CPS communicates with the local loader on behalf of its component

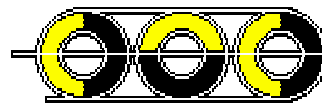


PERCO System for 2 GCF Models, C1 & C2



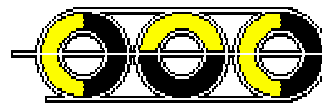
Modus Operandi

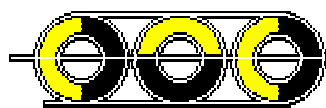
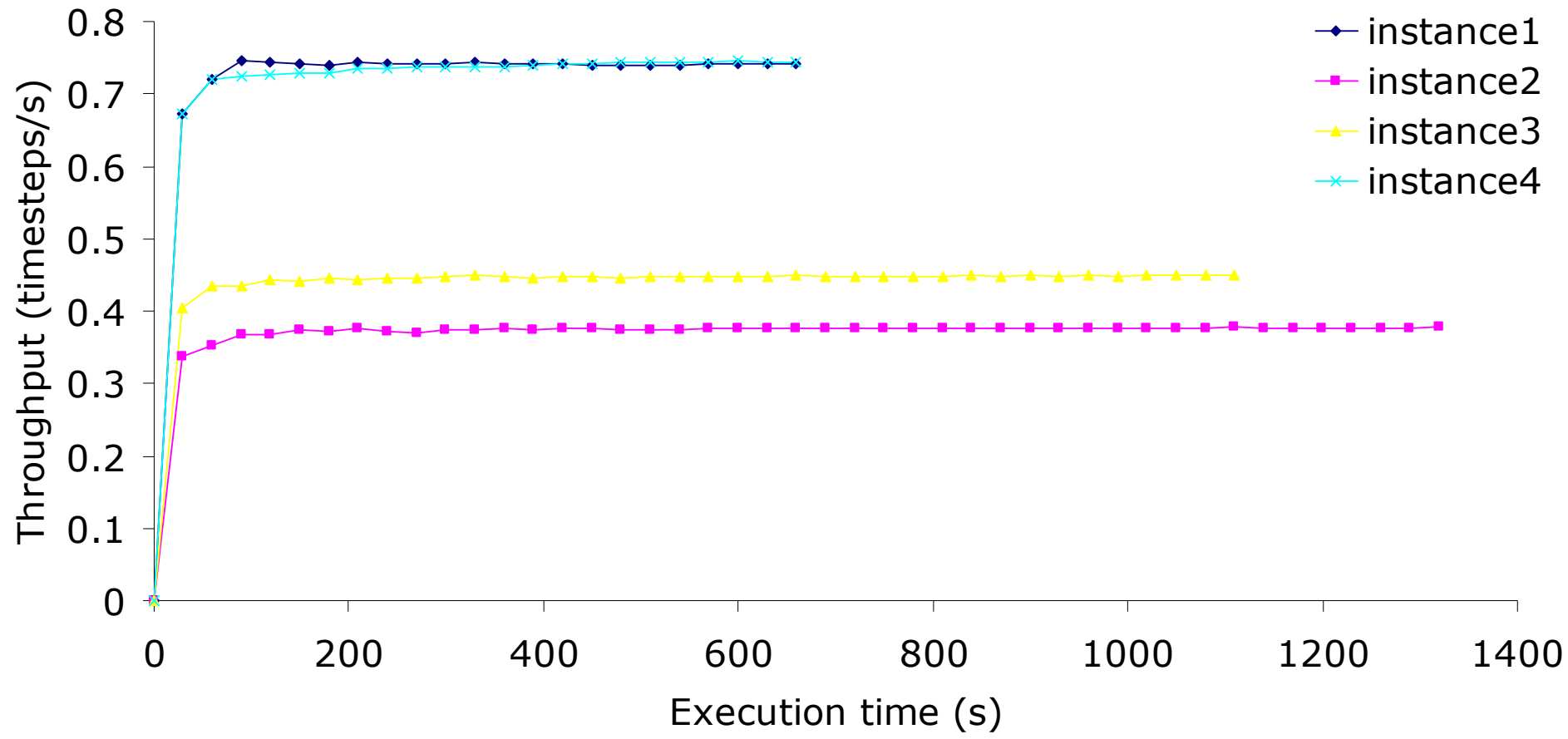
- At the k^{th} safe-point, the APS compares possible remaining execution times, which are $(K-k)$ times
 - the predicted times for all possible deployments
 - the most recently measured time on the current platform for a single major cycle, and then, according to the situation, the APS may ask each CPS to:
 - **Deploy** ($k=0$) models + CPSs
 - **Redeploy** ($K>k>0$) model(s) + CPS(s)
 - **Continue** ($K>k>0$) to use current deployment
 - **Terminate** ($k=K$)
- if models are redeployed, the cost of redeploying is estimated, and an optimum deployment among those possible is selected (big issue!)

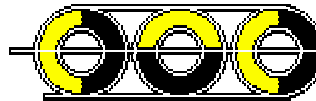
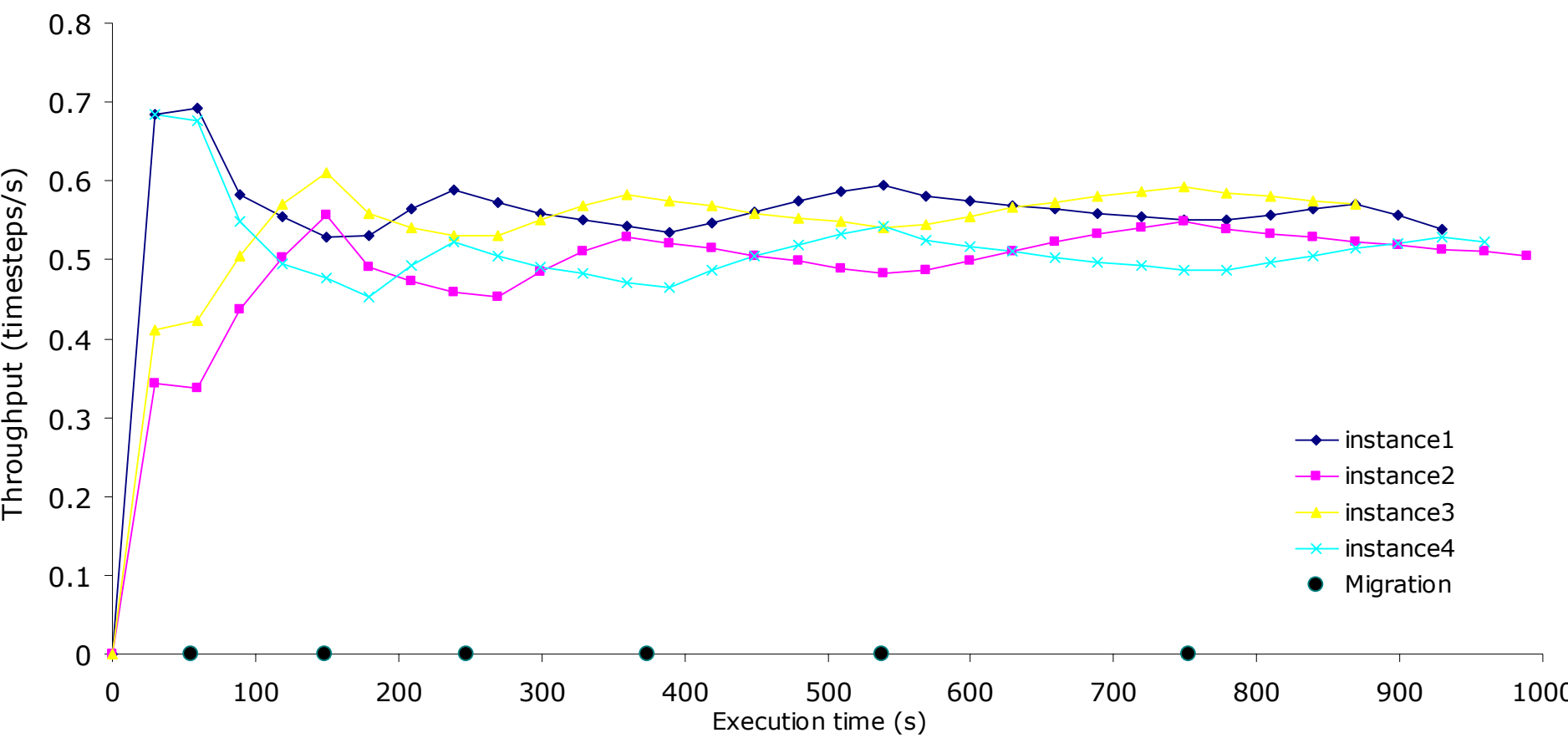


Prediction for a Major Cycle

- Based on the coupled model description, plus some data about expected execution times for the various models on various platforms, it is feasible to estimate the execution time for one major group given a specific deployment of models on platforms
- Could search for optimum (re-)deployment by predicting this time for all possible deployments (but probably better to use some form of non-exhaustive search!)

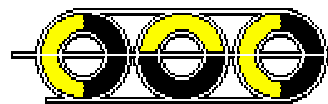






Current Status of PERCO

- demonstrated:
 - initial deployment
 - communications between APS and CPS
 - communications between loaders
 - migration of components between platforms
 - communications between components
 - repository for prediction data (inter-platform transfers and minor cycle execution times)
 - performance control of a simple assemblage of components
- in progress:
 - (efficient) communications between components
 - performance prediction capability
 - heuristic to reduce search space for (re)-deployment



Summary

- We are investigating the practicalities of component-based performance control in Grid execution environments
- We have shown that meaningful component-based software can be generated from scientific coupled models expressed in GCF notation
- A prototype performance control system is under development and we are nearly ready to apply it to a realistic scientific application

C. Armstrong, R.W. Ford, J.R. Gurd, M. Luján, K.R. Mayes, G.D. Riley, **“Performance control of scientific coupled models in Grid environments”, environments**, *Concurrency and Computation: Practice and Experience* (to appear).

K.R. Mayes, M. Luján, G.R. Riley, J. Chin, P.V. Coveney and J.R. Gurd

“Toward Performance Control on the Grid”, *Royal Society Philosophical Transactions A* (submitted).



Related Projects at Manchester

- Met Office - FLUME - design of next generation Unified Model software
 - Coupled models
- Tyndall Centre - SoftIAM - Climate Impact, Integrated assessment modelling
 - Coupling climate and economic models
- Computational Markets
 - UK e-Science funded (Imperial College led)

For more information check
<http://www.cs.man.ac.uk/cnc>

