
Implementing Feedback Control for Efficient Autonomic Solutions on the Grid

Rizos Sakellariou

University of Manchester

The vision of autonomic computing

(From <http://www.research.ibm.com/autonomic/index.html>)

"Civilization advances by extending the number of important operations which we can perform without thinking about them." (Alfred North Whitehead, 1900)

Observation: the complexity of current software systems requires (expensive) human intervention. Still, more interconnectivity (and complexity) is envisaged.

The idea: build computing systems that regulate themselves (self-management) pretty much in the same way our autonomic nervous system regulates and protects our bodies. (Paul Horn, October 2001)

(see IEEE Computer, January 2003, The Vision of Autonomic Computing)

Self-Management

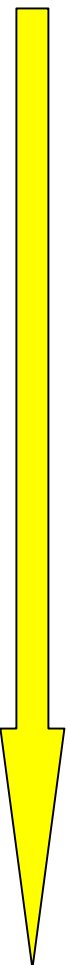
- Self-* properties:
 - Self-configuration (it is assumed that autonomic computing is only about this – wrong!)
 - Self-healing, self-protection, ...
 - **Self-optimization**: “*Components and systems continually seek opportunities to improve their own performance and efficiency*” (IEEE Computer 2003). I would use **adaptivity**:
 - Often mentioned (about 4M entries in Google Scholar)
 - Equally often, only *ad hoc* (not generic) solutions exist

An Example: queues are evil, but...



- Human beings can self-adapt and avoid long queues
- In a computing system the queue size may keep growing...

Timeline



1998

1998-2001: Parallel Object
Databases (EPSRC; Man/Ncl)

Grid era...

2001-2004: High Performance Query
Processing on the Grid (EPSRC; Man/Ncl)

2002-2006: OGSA-DAI: Grid Database
Access (e-Science; Edi/IBM/Man/Ncl)

OGSA-DAI

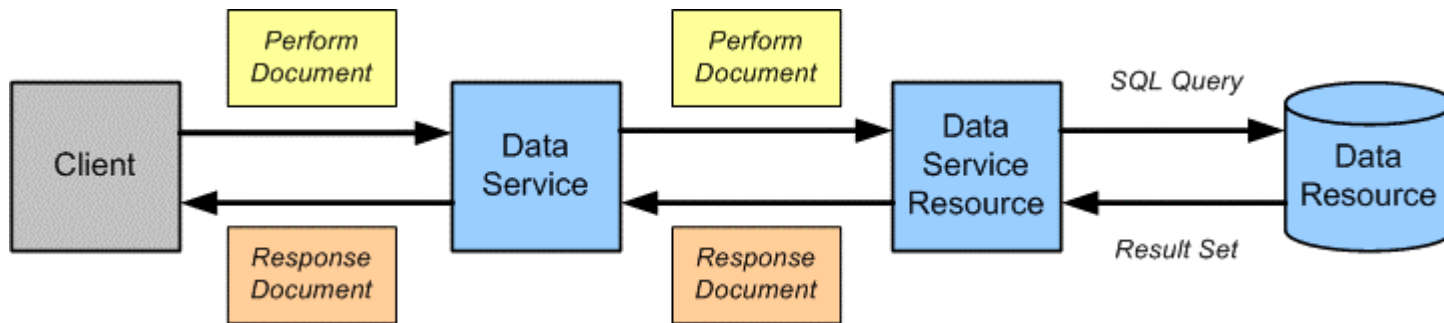
OGSA-DQP

2006-2009: Adaptive Systems Development
(EPSRC; Man/Ncl)

High-Performance
Computing,
Performance
Optimization,
Performance Prediction
Scheduling, etc...

OGSA-DAI in brief

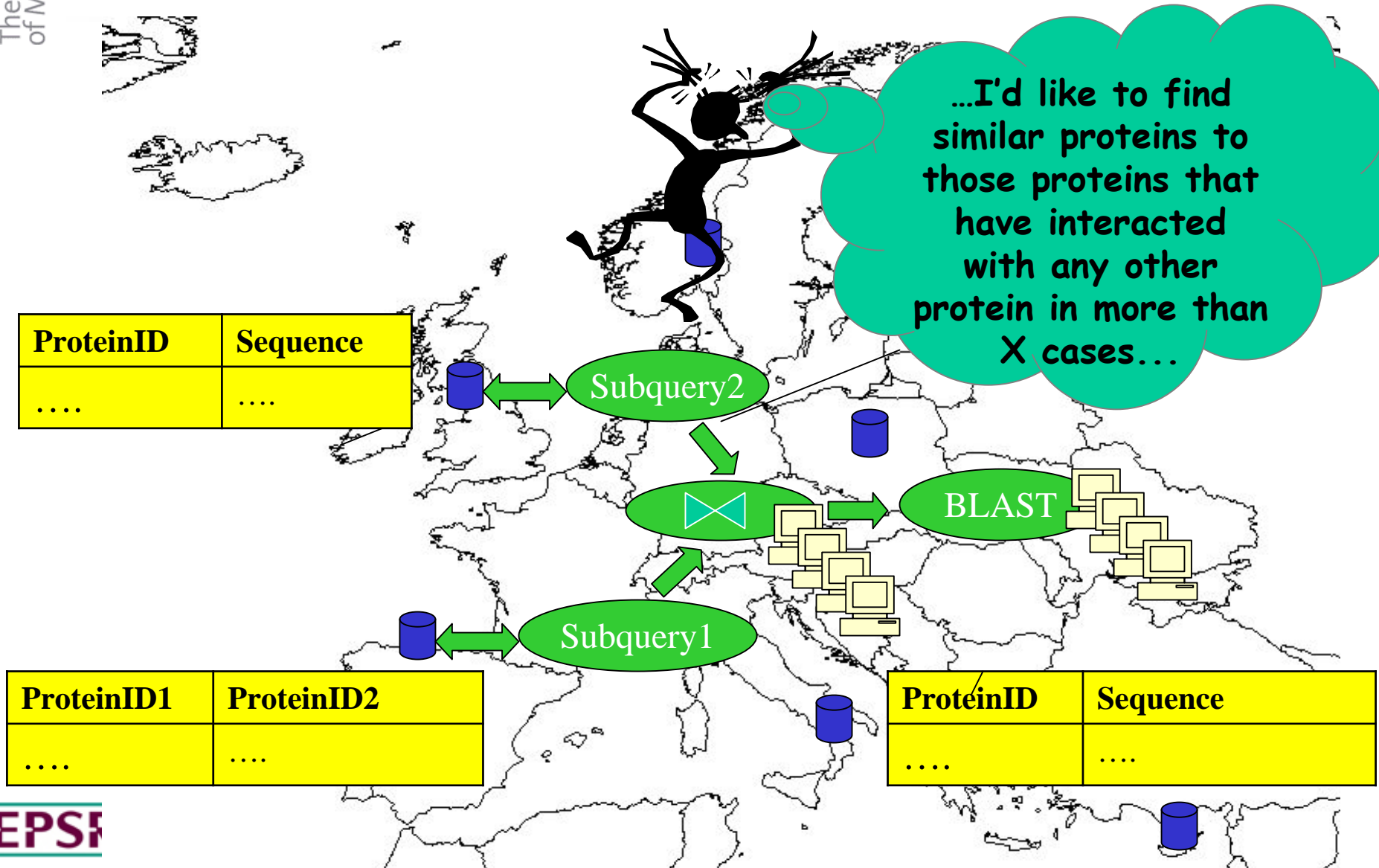
- Different types of data resources - including relational, XML and files - can be exposed via web services onto Grids.
- A number of popular data resource products are supported.
- Through OGSA-DAI, data resources can become grid-enabled in a clean, non-intrusive, straightforward way.



- “*Perform*” documents can describe/include data creation, query, modification and delivery tasks

For more information, visit the OGSA-DAI website: <http://www.ogsadai.org.uk>

Services For Accessing Multiple Grid Data Sources and Computational Resources (OGSA-DQP)

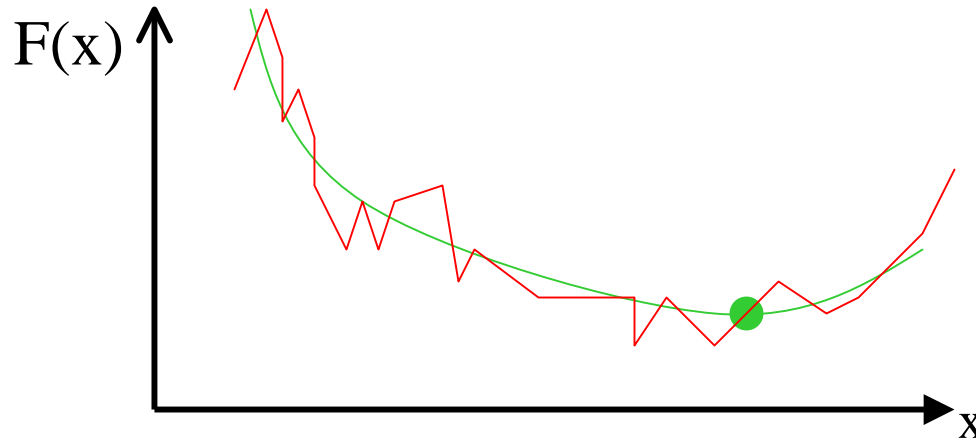


Key observations

- OGSA-DQP delivered functionality but good performance was a different cup of tea.
- There was an obvious need for **adaptive solutions**:
 - not only to respond to environment changes
 - but, also to tune different parameters of the middleware (**this is not trivial**)
- This is a **self-optimization** challenge!

Service tuning is non-trivial

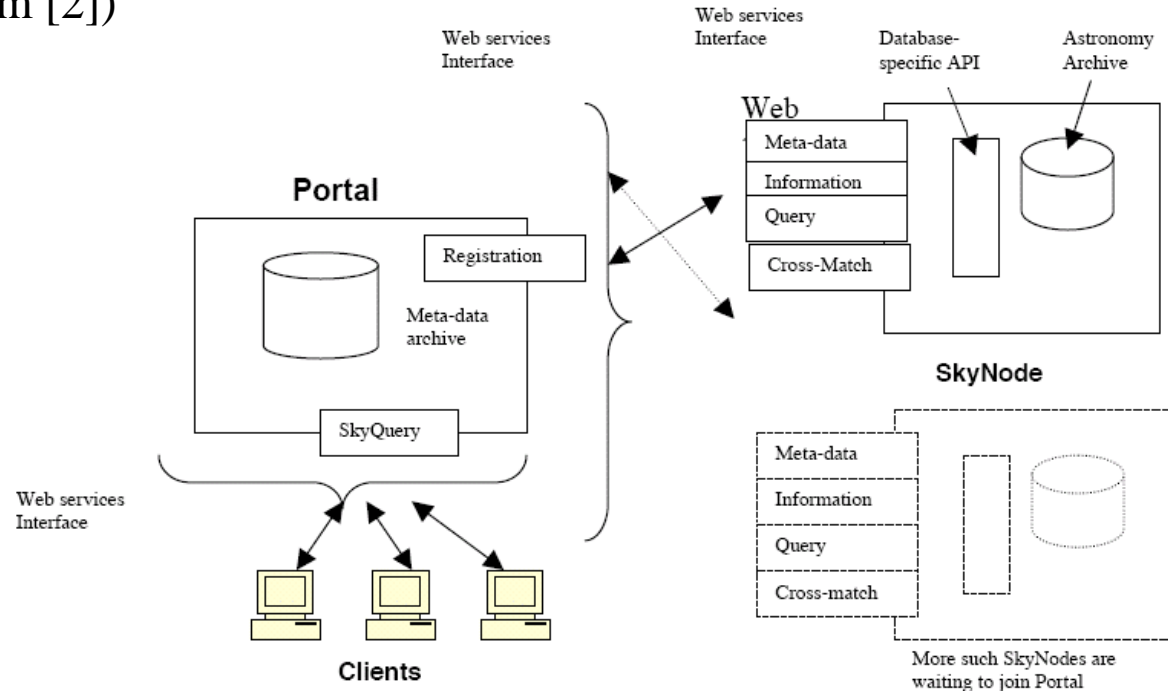
- Concave-like performance graphs are very common



- BUT, finding the optimal point is challenging, since the volatility of the environment and the noise insert local optima and non-linearities
- In addition, we may have functions of the form $F(x,y,z,\dots)$ instead of $F(x)$

Web Services (WSs) play an increasingly important role in large-scale data management applications

- E.g., SkyQuery (figure taken from [2])



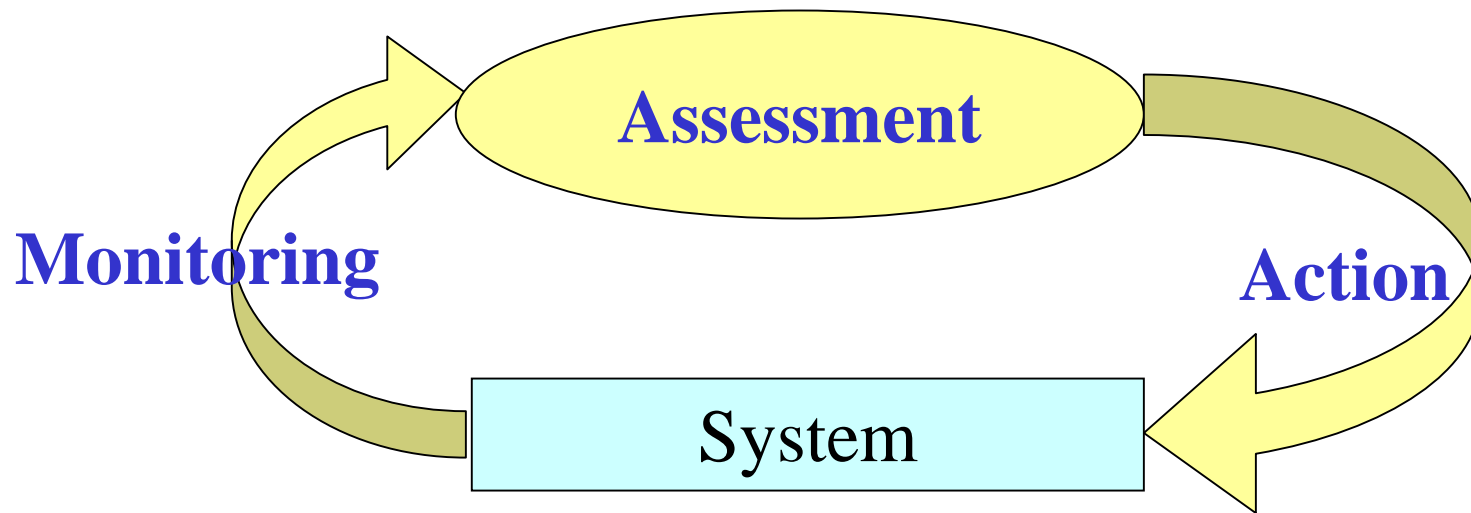
- Or more generic WS Management Systems that compile, optimize and evaluate workflows composed of multiple calls to remote WSs in order to analyze data [1]

[1] U. Srivastava, K. Munagala, J. Widom, and R. Motwani. Query optimization over web services. *VLDB*, 2006.

[2] Tanu Malik et al. SkyQuery: A Web Service Approach to Federate Databases. In *CIDR 2003*.

Self-Optimization

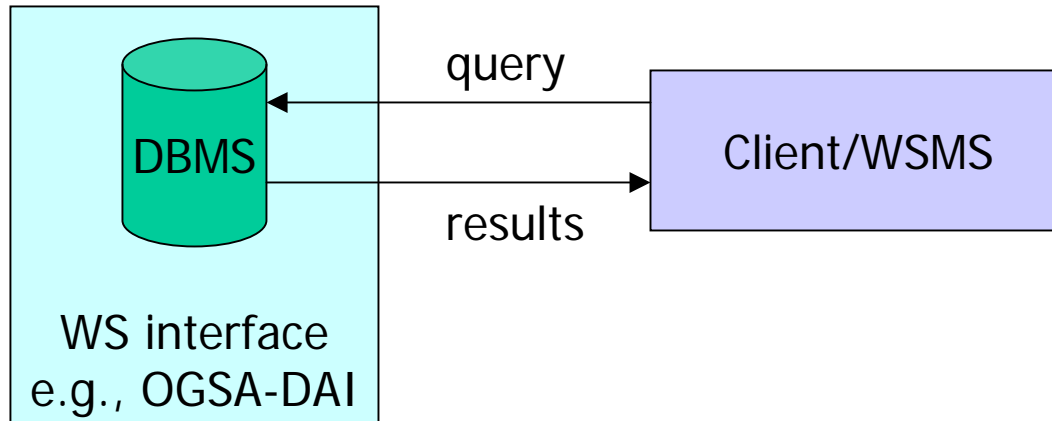
- “Components and systems continually seek opportunities to improve their own performance and efficiency” (IEEE Computer 2003)
- This implies:



Feedback Control

- How to implement this loop in a generic way?
- What are the right mechanisms to perform the assessment?
- Case study 1: Dynamic selection of block size in OGSA-DAI using ***control theory***.
- Case study 2: Adaptive workflow execution with the Pegasus workflow management system using ***utility functions***.

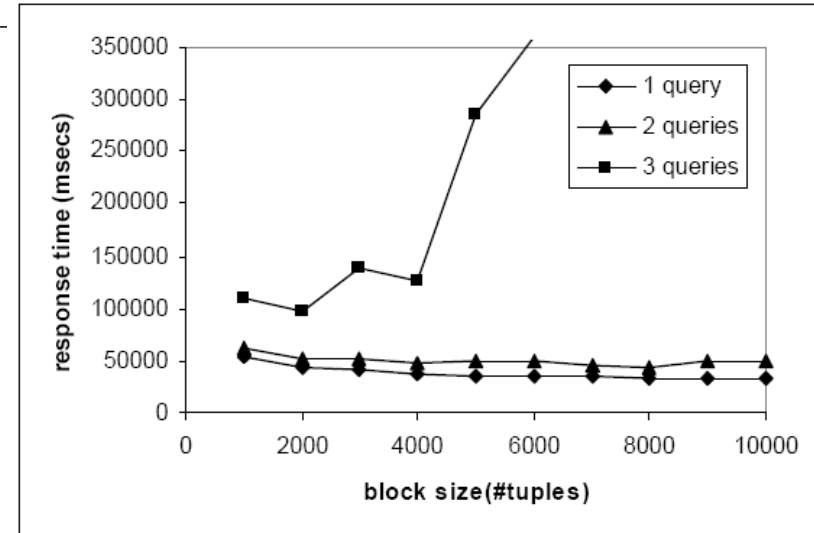
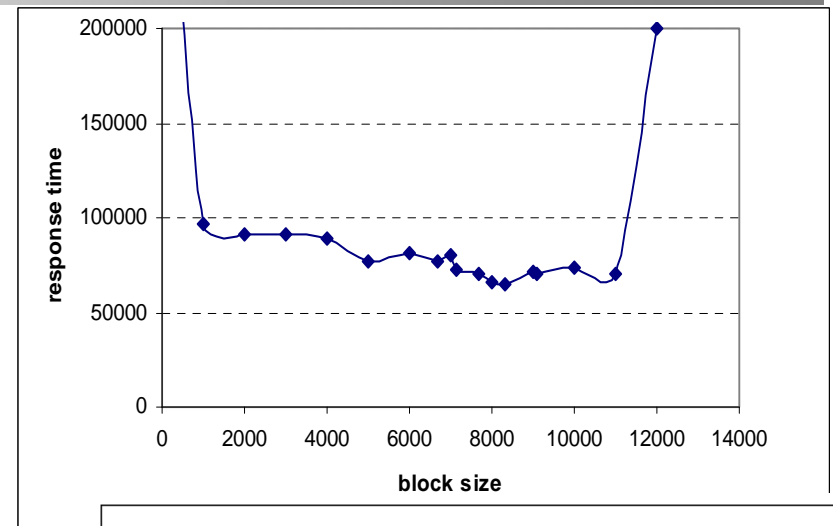
Motivation



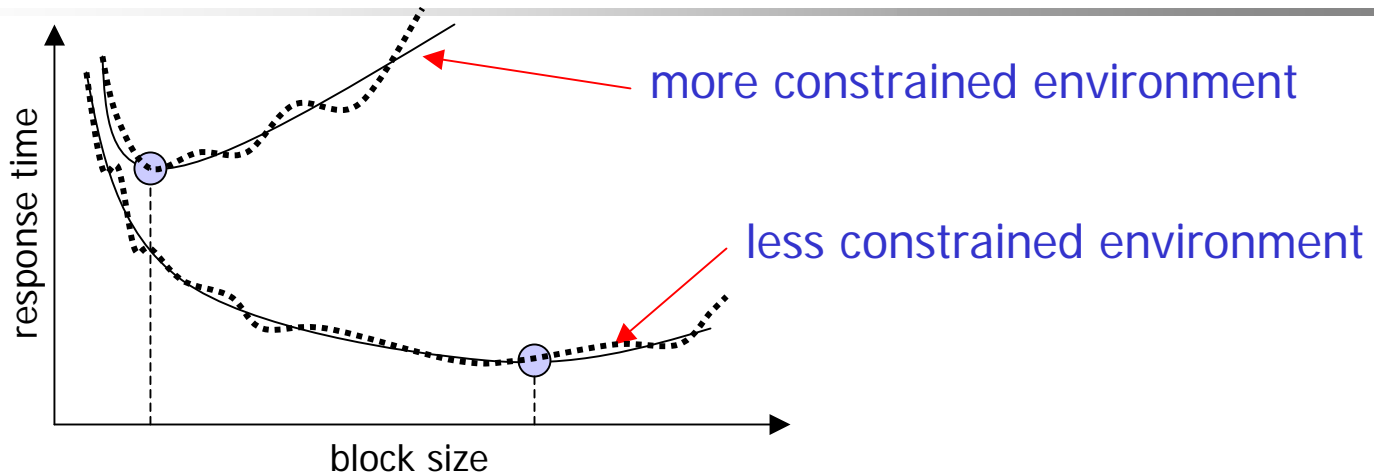
- WSs are slow (parsing overheads, protocols).
- Typically, the dominant cost is due to data transfer.
- Splitting the result set into blocks helps reduce this cost.

A real problem

- We want to retrieve a large dataset from a WS
 - Communication cost dominates
 - Typical scenario in OGSA-DAI/DQP
- It is widely accepted that due to performance limitations of XML parsers, SOAP, etc., we must split the data into several chunks
- Indicative performance results are shown in the figures



Problem statement



Challenges in detecting the optimal point on the fly:

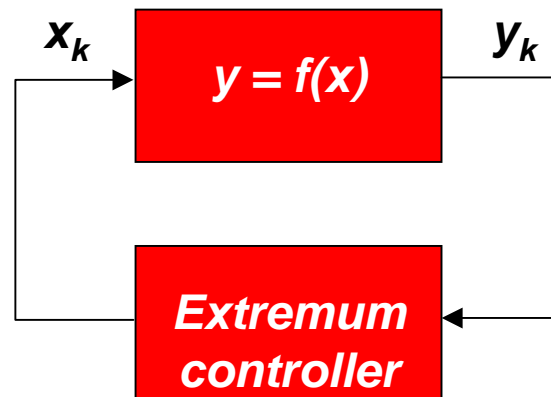
- **Volatility:**

- the optimal point changes with time even if the query is the same;
- the optimal point is different for different queries even if the connection/network conditions are the same;
- the optimal point can change even during the execution of the same query.

- **Absence of an analytical model**
- **Noise** (which causes local minima)
- **Requirement for fast convergence**

Control Theory 101

- **Objective:** find the (optimal) value of the control variable(s) that yields the maximum (minimum) value of a process output (or performance measure) in the presence of noise.



Runtime optimization

- Newton-based approach.
- Let x be the block size and $y=f(x)$ be the transfer time.
- One chunk transfer corresponds to one adaptivity step

$$x_k = x_{k-1} - \frac{\nabla f(x_{k-1})}{\nabla^2 f(x_{k-1})}$$

$$\nabla f(k) \simeq \frac{\Delta y}{\Delta x} = \frac{y_k - y_{k-1}}{x_k - x_{k-1}}$$

- Performance: LOW! (Such a technique is known to be sensitive to noise, and works better for quadratic models)

Switching extremum control

- Let y be the performance metric (that is, response time) and x_k the block size at the k th step. Then

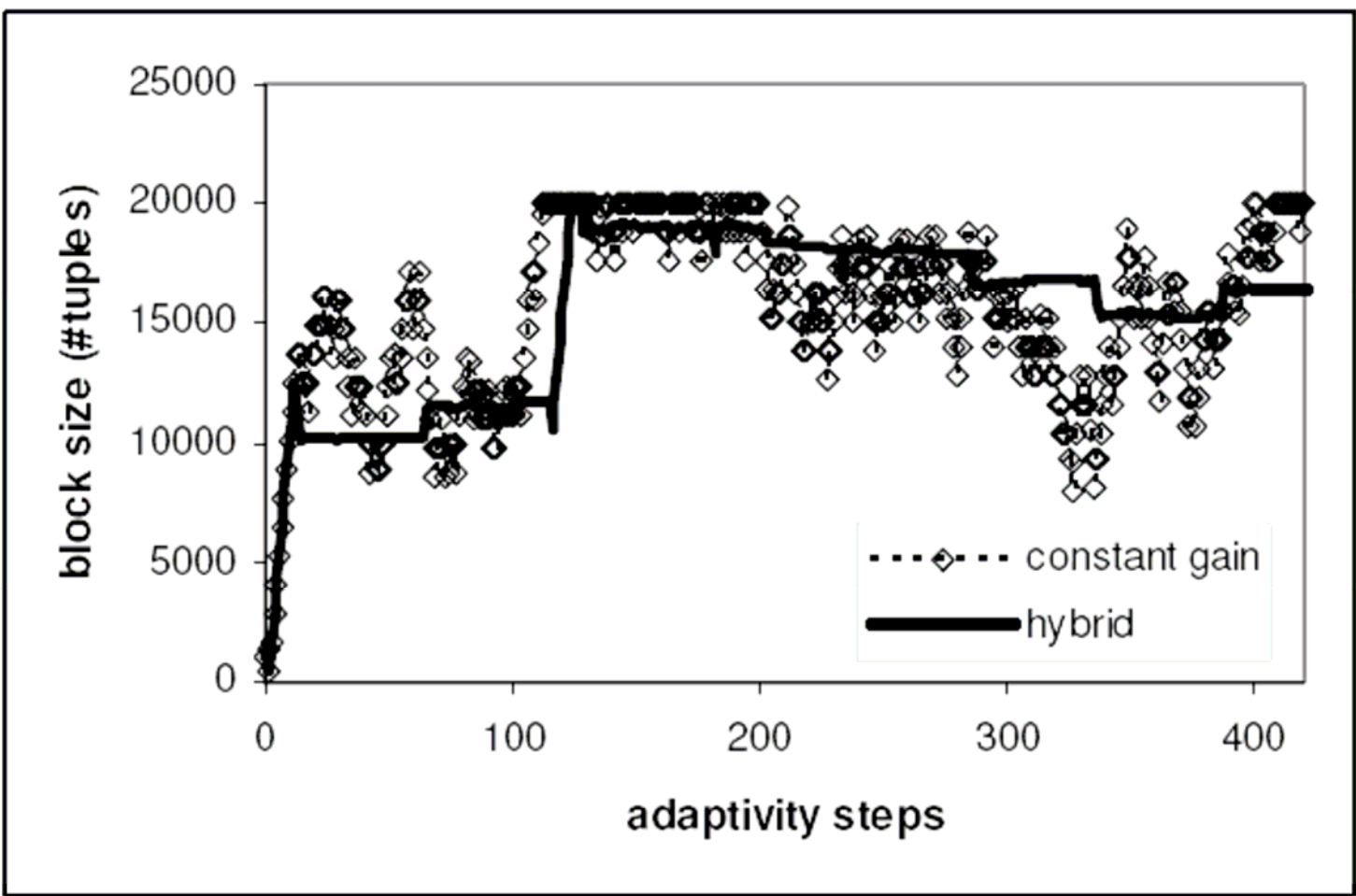
$$x_k = x_{k-1} - \text{gain} * \text{sign}(\Delta x * \Delta y)$$

- Rationale:
 - detect the side of the optimum point where the current block size resides on.
- Intuition behind:
 - the next block size must be greater than the previous one, if, in the last step, an increase has led to performance improvement, or a decrease has led to performance degradation.
 - Otherwise, the block size must become smaller.
- Different options exist for the choice of *gain*.

Lessons learned from experiments...

- Extensive experimentation using nodes in UK, Cyprus, Greece, different types of queries, models for machine load, etc.
- Different adaptivity policies have performance differences which are not negligible. Switching Extremum Control performs better than Newton techniques (as expected). No policy outperforms the other policies consistently.
- If the near optimal region is unknown and our static choice is conservative, improvements can be higher than 40% (even by an order of magnitude)
- Dynamically adjusting the block size outperforms fixed size configurations by more than 4% on average, even if these are known, e.g., through profiling, and set to the estimated optimum before execution.
- Oscillation and overshooting may occasionally become a problem.
- Switching between a number of controllers (hybrid controllers) leads to better results.

The behaviour may vary...



Control Theory: the good...

- A solid framework!
- Control theory gains popularity in autonomic/adaptive computing:
 - To meet QoS constraints
 - To dynamically tune systems
 - However, it is typically assumed that:
 - a model exists;
 - or, a model can be built through profiling
- Extremum controllers in computing
 - Error correction in unreliable networks
 - for dynamically configuring the number of redundancy packets included
 - Data transfer between WSs
 - Improvements to SOAP
- But...

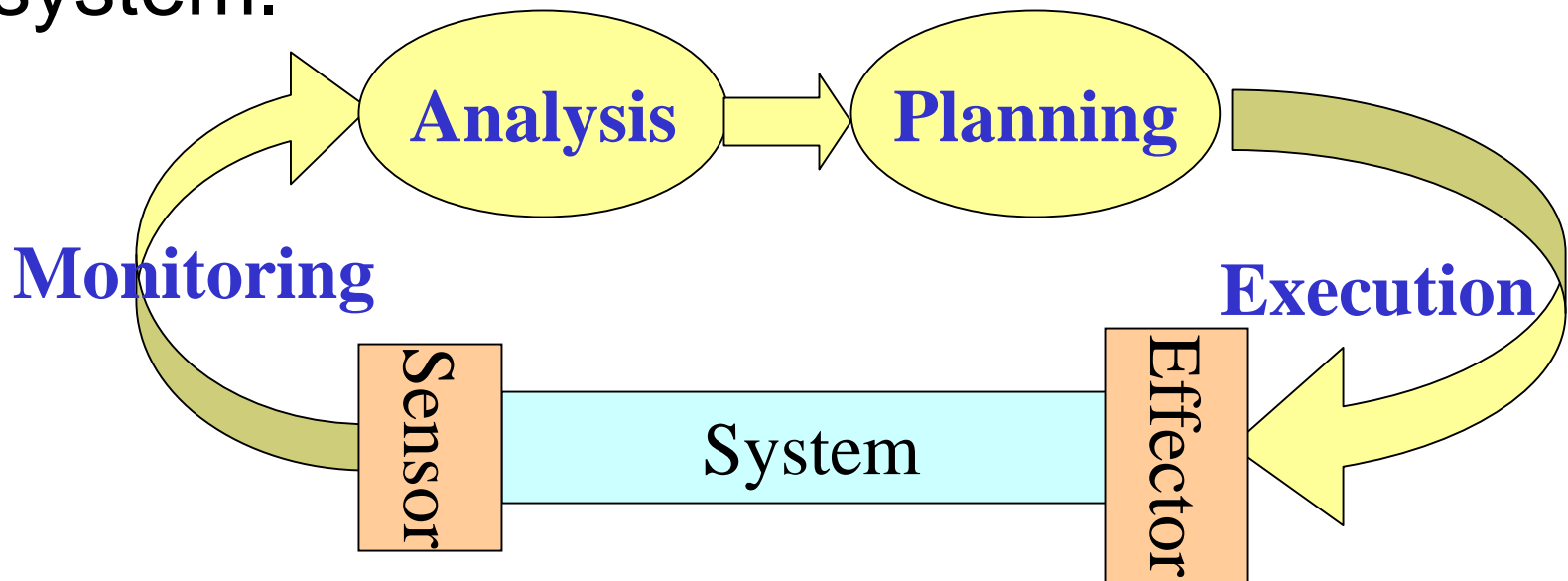
Control Theory: the bad...

Control Theory is useful when:

- the entity to be controlled exposes to the controller a tuning parameter which affects the performance metric:
 - not always the case (many parameters may also affect the performance metric in unknown ways)
- several measurements can be obtained and the impact of a suggestion to the controller is quickly assessed.
 - often, the impact of a choice is not known until some time later and new adaptations have a cost.
- *Control theory is not some kind of panacea!*
- **Other approaches for feedback control are needed – can they be incorporated within a generic infrastructure?**

An infrastructure to support adaptive systems development

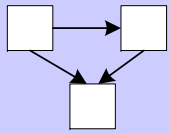
- Ease the development of adaptive systems
- Separate, as much as possible, the adaptive functionality from the rest of the system.



Workflow Composition and Execution in Pegasus

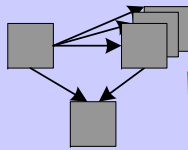
Workflow Evolution

Construct the Analysis



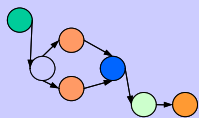
Workflow Template

Select the Input Data



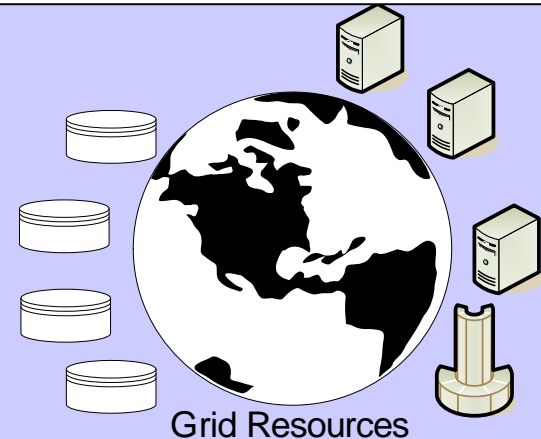
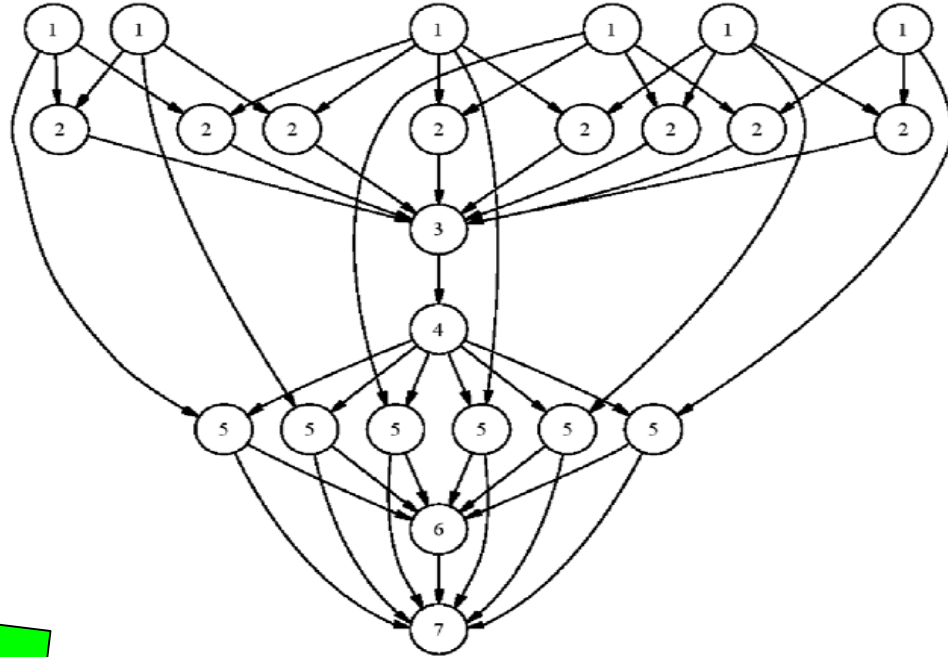
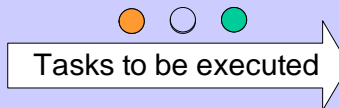
Workflow Instance

Map the Workflow onto Available Resources

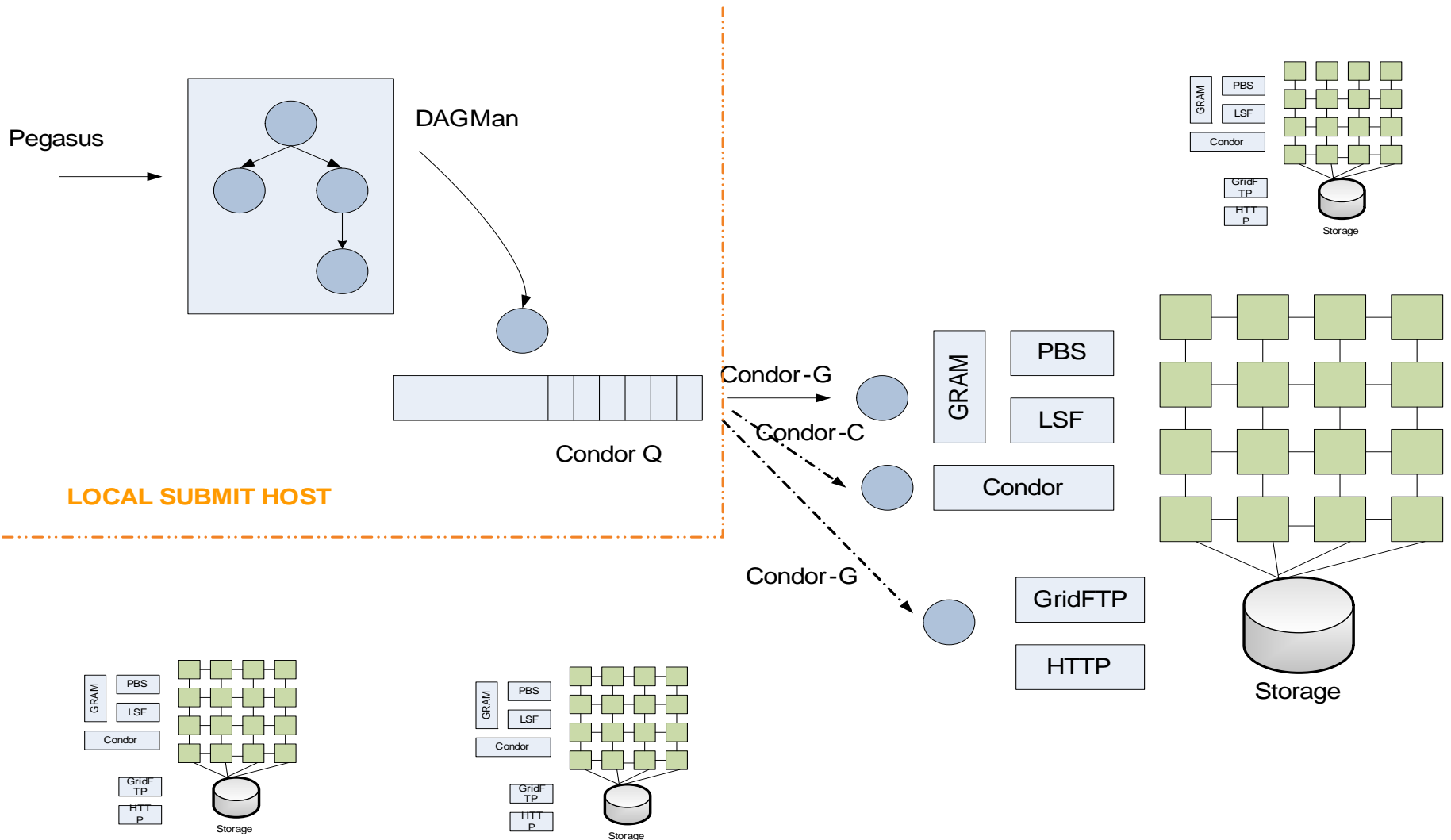


Executable Workflow

Execute the Workflow



Execution Environment



Case Study: Pegasus Workflow Management Environment

Characteristics of Pegasus workflow execution

- Very long running
- Small delays can have large effects due to dependencies
- Involve highly distributed resources
- Limited control over resources
- Uncertain execution times
- Uncertain queue waiting times
- Pegasus schedules a workflow before it starts executing
 - Using current information about the execution environment
- What happens if the environment changes?
 - Resources appear/disappear; the load changes

Adaptive Workflow Execution

Objective: adapt to site queue length, one of the biggest delays in execution, taking also into account deadlines for the completion of each workflow as well as the cost of the resources.

Monitoring: To monitor the progress of an executing workflow. Events: Job queue, Execute, Termination. Sensed from the Pegasus Log.

Analysis: Establish whether the workflow is performing according to expectations when it was compiled. Uses the CQL continuous query language to group and analyse the events produced by monitoring (CQL is SQL-like but with extensions for queries over time).

Planning: When analysis detects a sustained change in batch queue times for a site. Re-scheduling using a scheduler that takes into account historic data.

Execution: Halt the current workflow and deploy the newly planned one.

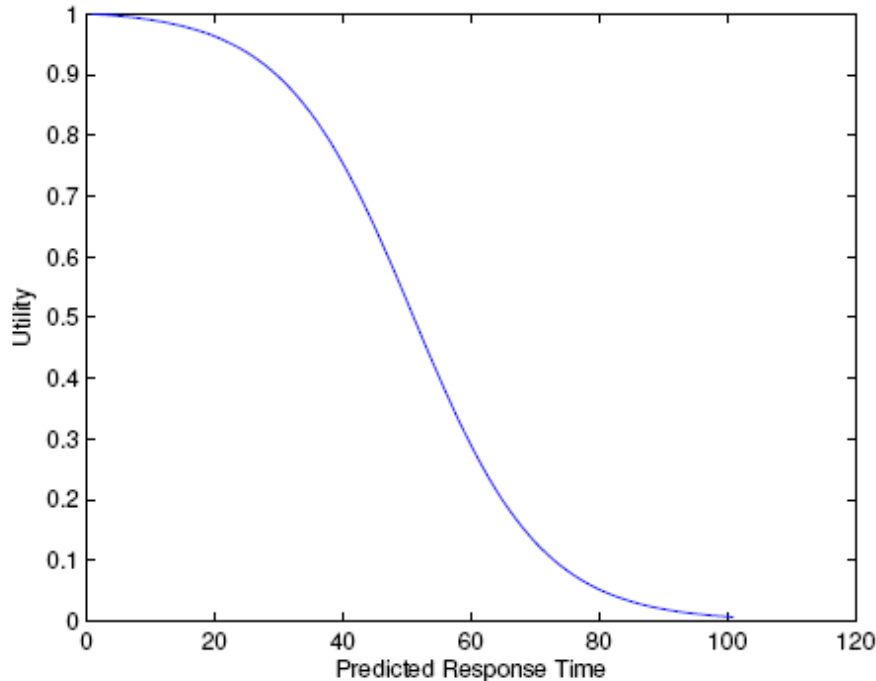
An Execution Scenario

- A provider provides computing resources; there is a charge for each resource (fastest machines are expected to cost more).
- Consumers want to run their workflows within a deadline. Different tasks of different workflows are added to the queue of each resource.
- The goal of the consumers is to complete the execution of the workflows within a deadline, but at the smallest possible cost.
- **Problem: what machines shall we choose?**

Results & Current Progress

- Retrofitting Pegasus with adaptive behaviour required minimum interference with Pegasus.
- The adaptive infrastructure is heavyweight (cf. control theory) but does not add a significant overhead.
- Experiments with synthetic workflows and Montage. Performance improvements of up to 50% were obtained but not consistently.
- Too few adaptivity steps (costly), too slow the response: not suitable for a control theory approach.
- **Feedback control**: decision making is based on modelling the problem with utility functions and then solve it as an optimization problem.

Utility-Driven Adaptations



K. Lee, et al. Utility Driven Adaptive Workflow Execution. CCGrid'09, to appear.

- For a given mapping, a utility function represents each goal (e.g., target response time, income, etc).
- These functions are treated as black-box functions, which are optimized using an implementation of a Mesh Adaptive Direct Search algorithm (MADS)

Lessons Learned

- Feedback control based on control theory is not useful in situations where the cost of adaptation is high and not too many measurements can be obtained (how high is ‘high’? how many is ‘too many’? – how can we decide what is suitable for a particular scenario?).
- In scenarios, such as workflow execution under constraints, utility functions can be used: once every parameter to be controlled is modelled by a utility, the problem of searching for a good solution can be solved using recently developed non-linear optimization algorithms.
- Scenarios such as the one assumed for workflow execution may dominate with the emergence of **Cloud Computing**.

To find out more...

Anastasios Gounaris, Christos Yfoulis, Rizos Sakellariou, Marios D. Dikaiakos: Self-optimizing block transfer in web service grids. *Proceedings of the 9th annual ACM international workshop on Web Information and Data Management (WIDM) 2007*, pp. 49-56

Anastasios Gounaris, Christos Yfoulis, Rizos Sakellariou, Marios D. Dikaiakos: A Control Theoretical Approach To Self-optimizing Block Transfer in Web Service Grids. *ACM Transactions on Autonomous and Adaptive Systems*, volume 3, issue 2, May 2008.

Anastasios Gounaris, Christos Yfoulis, Rizos Sakellariou, Marios D. Dikaiakos: Robust Runtime Optimization of Data Transfer in Queries over Web Services. *Proceedings of the 24th International Conference on Data Engineering (ICDE) 2008*, pp. 596-605

Kevin Lee, Rizos Sakellariou, Norman W. Paton, Alvaro A. A. Fernandes, Ewa Deelman, Gaurang Mehta. Adaptive Workflow Processing and Execution in Pegasus. *3rd International Workshop on Workflow Management and Applications in Grid Environments (WaGe'08)*, 2008.

Kevin Lee, Norman W. Paton, Rizos Sakellariou, Alvaro A. A. Fernandes. Utility Driven Adaptive Workflow Execution. *CCGrid'09*, to appear.

Summary

- There is a long way to go if we are to build generic adaptive / self-optimizing strategies.
- **Feedback control** needs to strike a balance between many aspects (e.g, how often will it be called, overhead, etc).
- Yet, there is a growing need for adaptivity (because of volatile environments and conflicting goals).
- **Grand Challenge**: *Think about the human brain. It evolves to a wonderful self-managing system from a single cell. To what extent can we do something similar with computing systems?*

The Persons

- *Anastasios Gounaris* (ex-Manchester, ex-Cyprus, now Thessaloniki)
- *Kevin Lee* (Manchester)
- *Christos Yfoulis* (Thessaloniki)
- Also: *Ewa Deelman* (USC/ISI), *Marios Dikaiakos* (Cyprus), *Alvaro Fernandes* (Manchester), *Gaurang Mehta* (USC/ISI), *Norman Paton* (Manchester), *Jim Smith* (Newcastle), *Paul Watson* (Newcastle)