

Adapting to Changing Resource Performance in Grid Query Processing

Anastasios Gounaris¹, Jim Smith², Norman W. Paton¹,
Rizos Sakellariou¹, Alvaro A.A. Fernandes¹, Paul Watson²

¹University of Manchester, UK

²University of Newcastle, UK



Outline of Talk

- Query Processing on the Grid with OGSA-DQP
- A generic framework for adaptive query processing (AQP)
- Adaptations to Changing Resources in Grid query processing
 - Design
 - Evaluation

Selecting Resources in OGSA-DQP

The screenshot displays the DQP Client - Administrator Mode interface. The Explorer panel on the left shows a tree view of resources, including Computational Nodes 1-5, DQP Service, DQP Factory, Registry, Data Resources, and Web Services. The Node Information panel in the center shows details for Node 4, such as Hosts Data Source (true), CPU Speed (1400.0 MHz), and Max Memory (262234 KB). The Schema panel on the right shows HTML tables for weighted_protein_sequence and protein_interaction, along with Imported Services and an Operation table.

resources

Node ID	4
Hosts Data Source	true
Hosts Service	false
Has Evaluator Factory	true
Node URI	130.88.198.202
CPU Speed (MHz)	1400.0
CPU Load Percentage	0.6
Max Memory (KB)	262234
Available Memory (KB)	120000
High Join Execution Times	13

name	length	sqlTypeName
ORF	50	varchar
sequence	65535	text
dummyfactor	12	float

name	length	sqlTypeName
ORF1	50	varchar
ORF2	50	varchar
baitProtein	50	varchar
interactionType	5	varchar
repeats	11	int
experimenter	100	varchar

Service Name	EntropyAnalyserService
Service Endpoint URI	http://phoebus.cs.man.ac.uk:9090/axis/services/EntropyAnalyserService
Service Namespace	http://phoebus.cs.man.ac.uk:9090/axis/services/EntropyAnalyserService

Operation	Arguments	Return Type
calculateEntropySlow	(stSequence : string)	double
calculateEntropy	(stSequence : string)	double

Unified schema

Evaluating Queries in OGSA-DQP

query

The screenshot displays the DQP Client interface. The 'Query Expression' window contains the following SQL query:

```
select go.name, p. ORF, i. ORF2
from p in protein_goters, i in protein_interactions,
go in goterms
where p.GOTermIdentifier=go.id and p. ORF=i. ORF1
and p. ORF like "YBL0%";
```

The 'Join_8 Results' window shows a table with the following data:

goterm.name	protein_goterm. ORF	protein_interaction. ORF2
DNA binding	YBL002W	YCL066W
nucleosome	YBL002W	YCL066W
chromatin assembly/disassembly	YBL002W	YCL066W
cytoplasm	YBL016W	YDR469W
protein amino acid phosphorylation	YBL016W	YDR469W
cell cycle arrest	YBL016W	YDR469W
cytoplasm	YBL016W	YDR480W
protein amino acid phosphorylation	YBL016W	YDR480W
cell cycle arrest	YBL016W	YDR480W
cytoplasm	YBL016W	YHR168W
protein amino acid phosphorylation	YBL016W	YHR168W
cell cycle arrest	YBL016W	YHR168W
cytoplasm	YBL016W	YIL169C
protein amino acid phosphorylation	YBL016W	YIL169C
cell cycle arrest	YBL016W	YIL169C
cytoplasm	YBL016W	YPL049C
protein amino acid phosphorylation	YBL016W	YPL049C
cell cycle arrest	YBL016W	YPL049C
chromatin binding	YBL023C	YOR196C

The 'Join_8 Plan' window shows a query execution plan diagram. The plan starts with three table scans: 'TABLE_SCAN (3)' for 'protein_interaction', 'TABLE_SCAN (2)' for 'protein_goterm', and 'TABLE_SCAN (1)' for 'goterm'. These are joined by 'EXCHANGE (3) -> (2, 3)', 'EXCHANGE (2) -> (2, 3)', and 'EXCHANGE (1) -> (1, 2, 3)' respectively. The results are then processed by 'HASH_JOIN (2, 3)' and 'HASH_JOIN (1, 2, 3)' operators, followed by 'PROJECT (3)', 'PROJECT (2)', and 'PROJECT (1)' operators. The final output is produced by a 'DELIVER (0)' operator.

plan

Service Interactions in OGSA-DQP

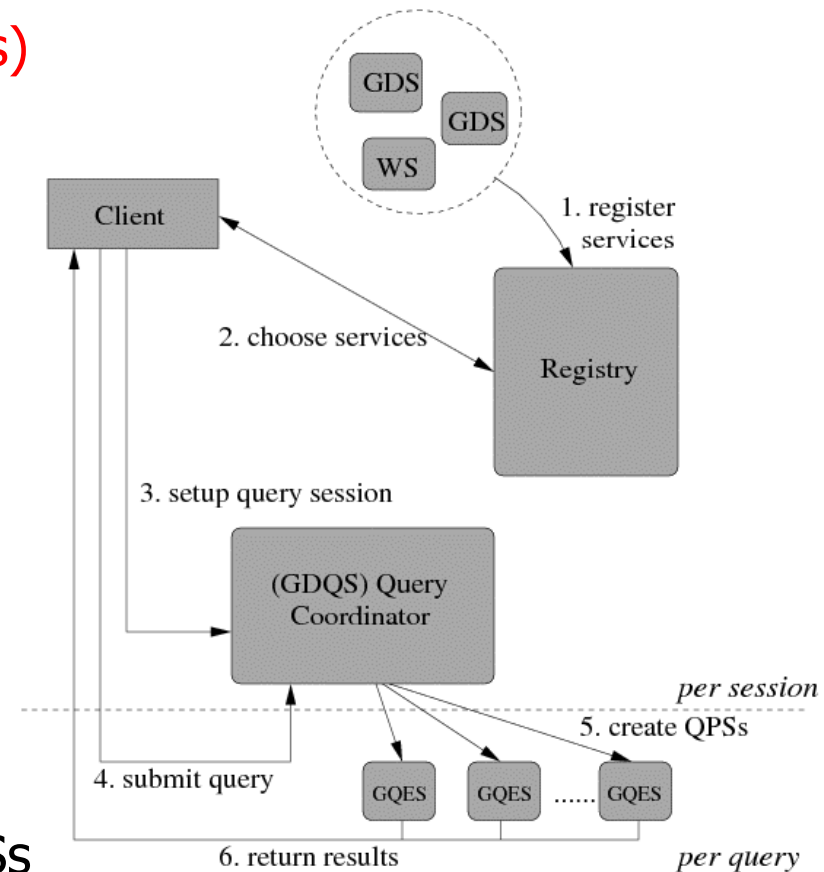
Grid Distributed Query Services (GDQSs)

that:

- parse, compile, partition and schedule the query execution over a union of distributed data sources.
- Coordinates the GQESs into executing the plan

Grid Query Evaluation Services (GQESs) that:

- implement the physical query algebra;
- run a partition of a query execution plan generated by a GDQS;
- interact with other GQESs/GDSs/WSs





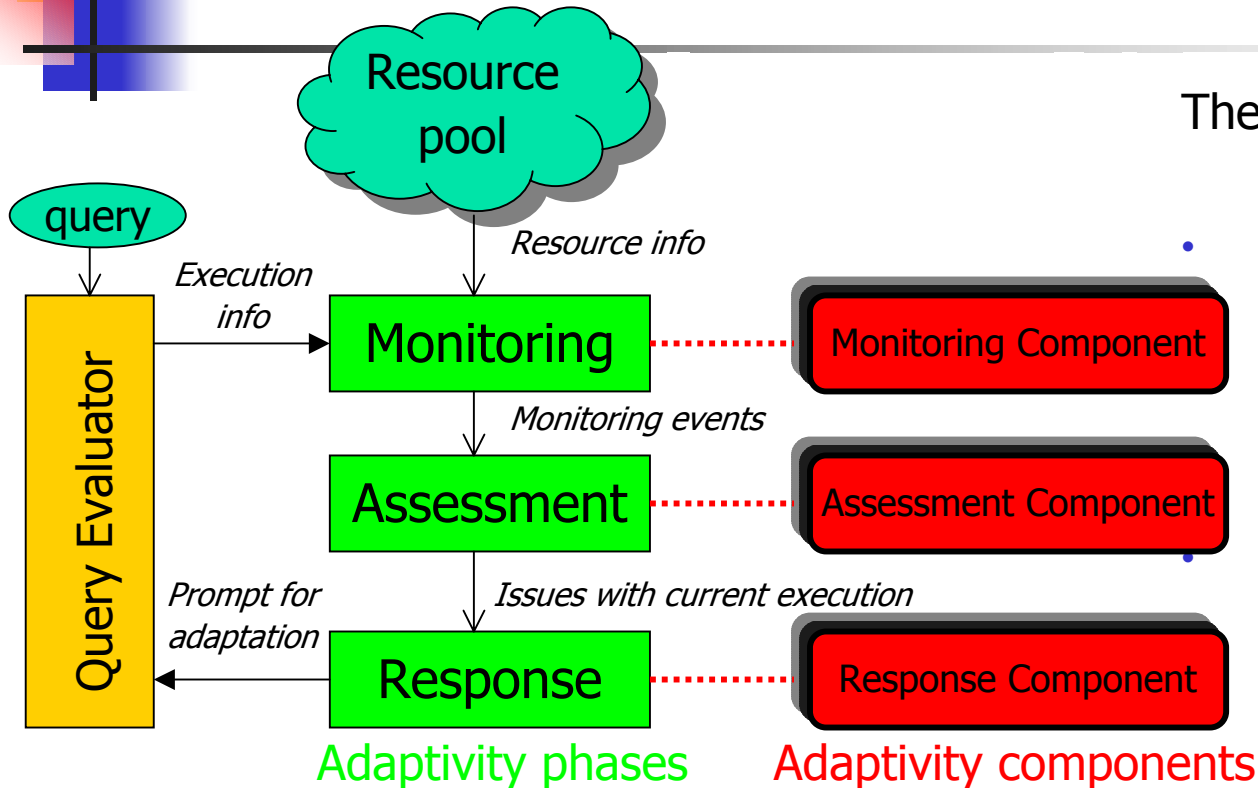
Limitations of existing AQP systems:

- They are too specific in terms of the problem they address and are designed in isolation.
 - No interoperability.
 - Many different, non-compatible solutions for the same problem.

Z. Ives, A. Halevy, and D. Weld. Adapting to source properties in processing data integration queries. In *Proc. of the 2004 ACM SIGMOD*.

A. Gounaris, N.W. Paton, A. A. A. Fernandes, and R. Sakellariou. Adaptive query processing: A survey. In *19th BNCOD*, Springer, 2002.

Adaptivity Framework



The framework is a step toward:

- Development of accepted abstractions.
- Generic design methodology for AQP systems.
- Reusable adaptivity components.

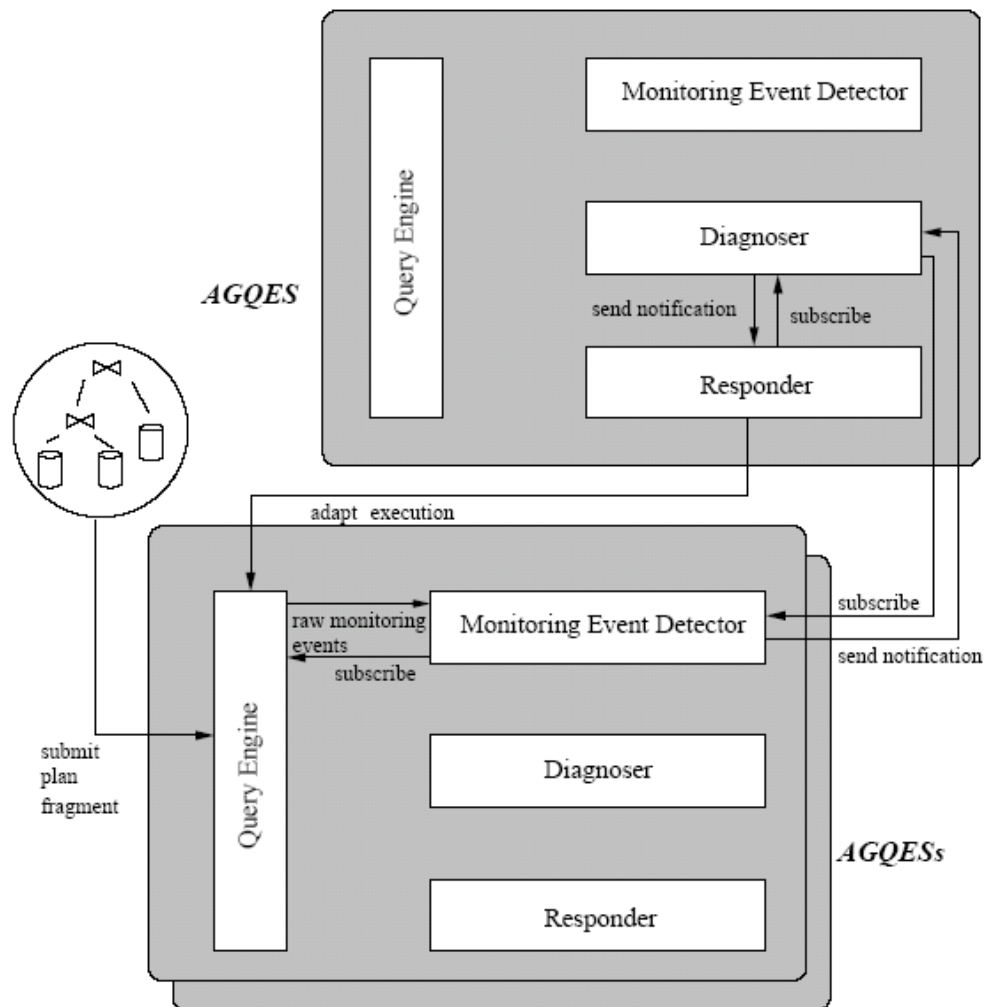
Current techniques tend to group together an approach to monitoring, a means of assessment, and a form of response.



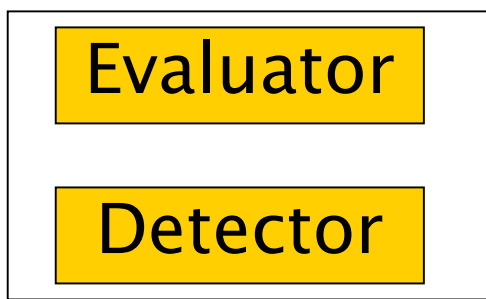
Outline of Talk

- Query Processing on the Grid with OGSA-DQP
- A generic framework for adaptive query processing (AQP)
- **Adaptations to Changing Resources in Grid query processing**
 - Design
 - Evaluation

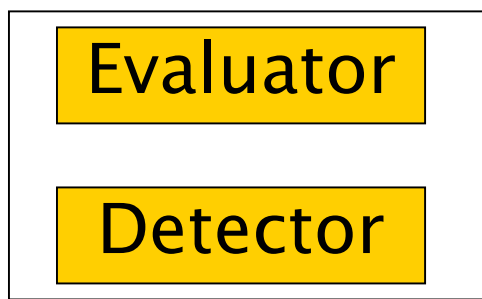
Adapting to workload imbalance: Architectural Components



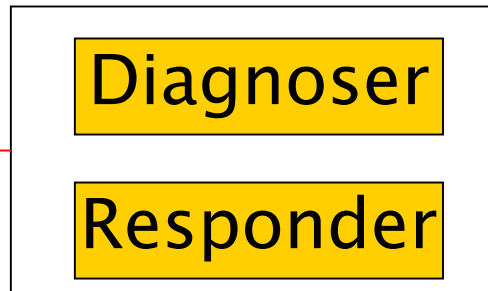
Instantiating the framework: a dynamic balancing example



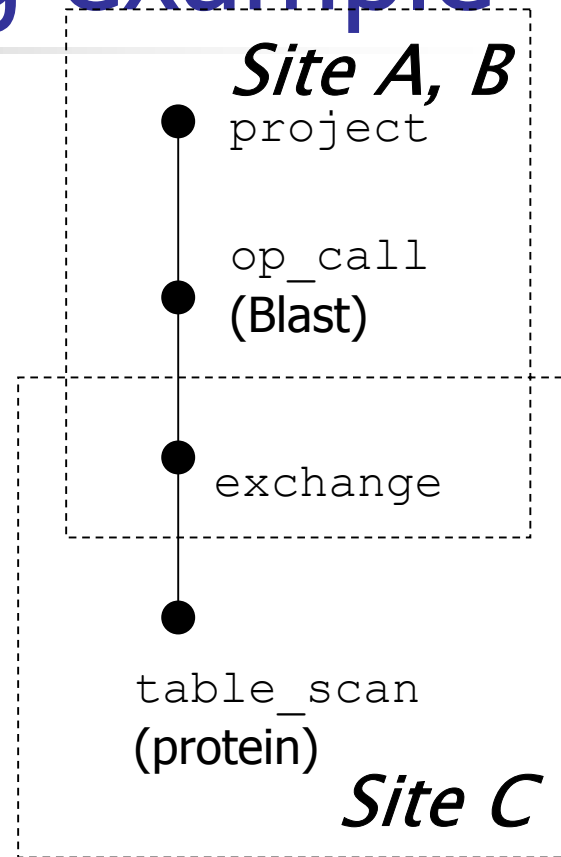
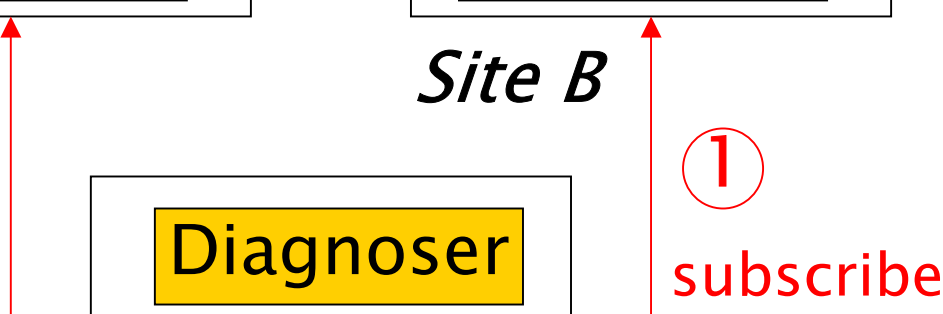
Site A



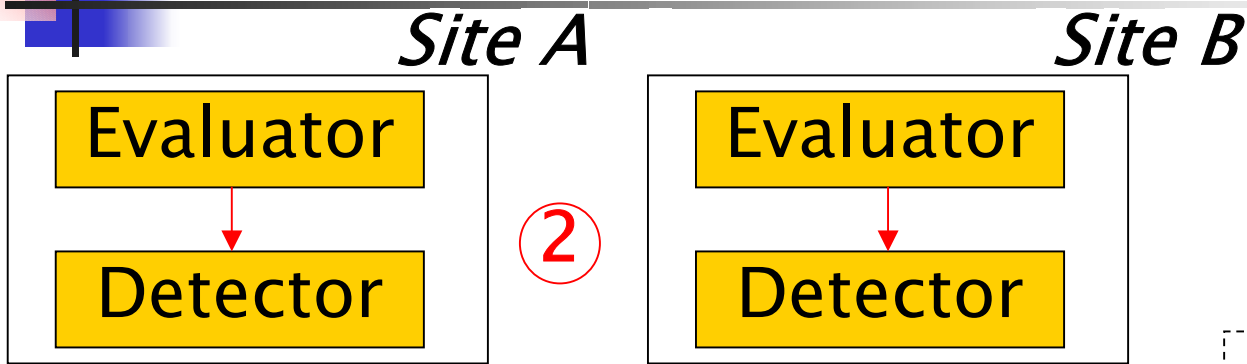
Site B



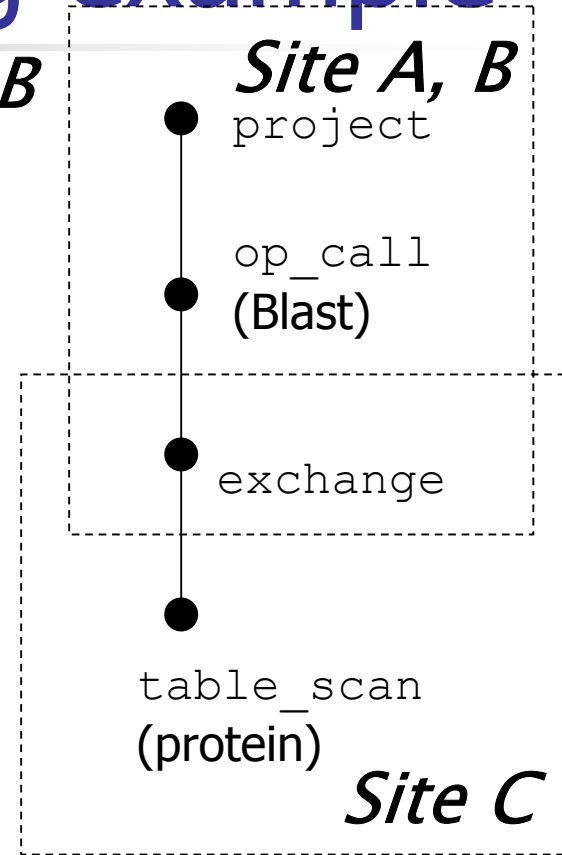
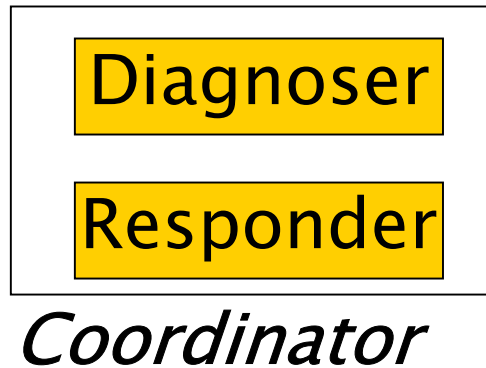
Coordinator



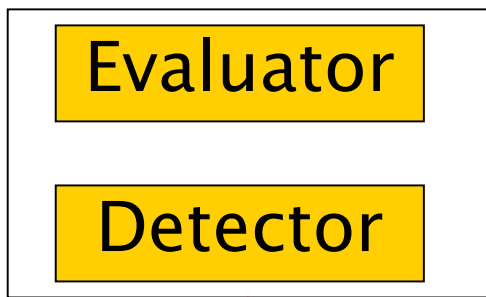
Instantiating the framework: a dynamic balancing example



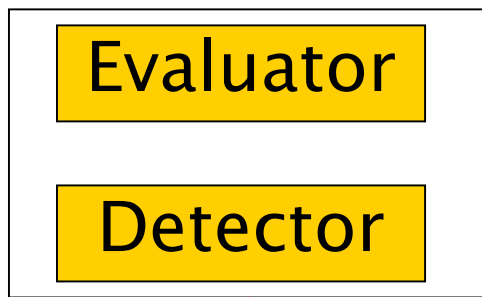
Evaluators pass on monitoring info to detectors



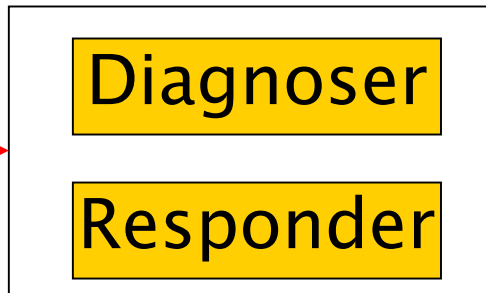
Instantiating the framework: a dynamic balancing example



Site A



Site B



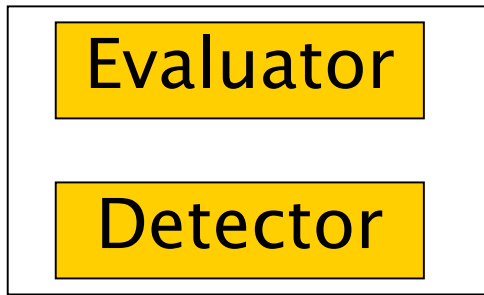
Coordinator

③

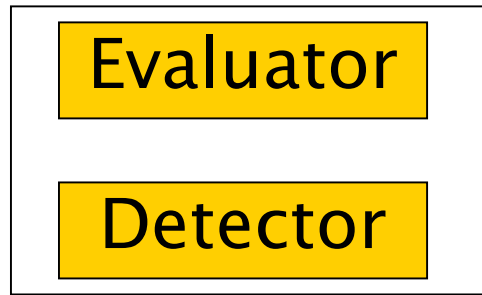
Detectors notify the diagnoser of the actual behaviour



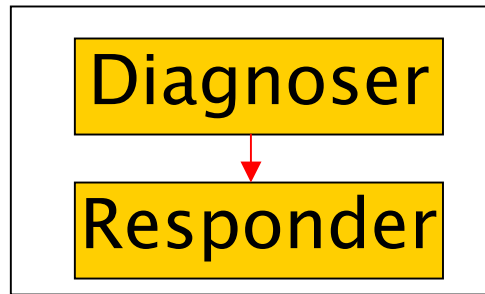
Instantiating the framework: a dynamic balancing example



Site A



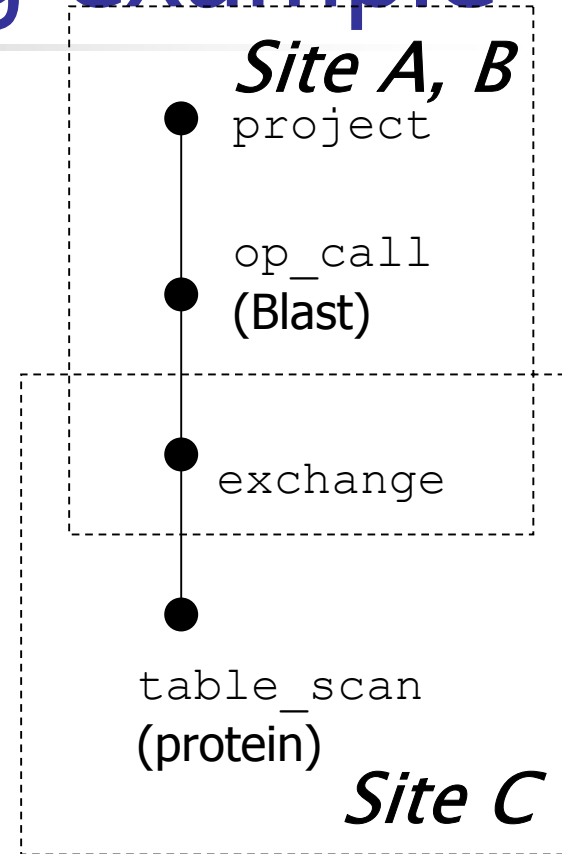
Site B



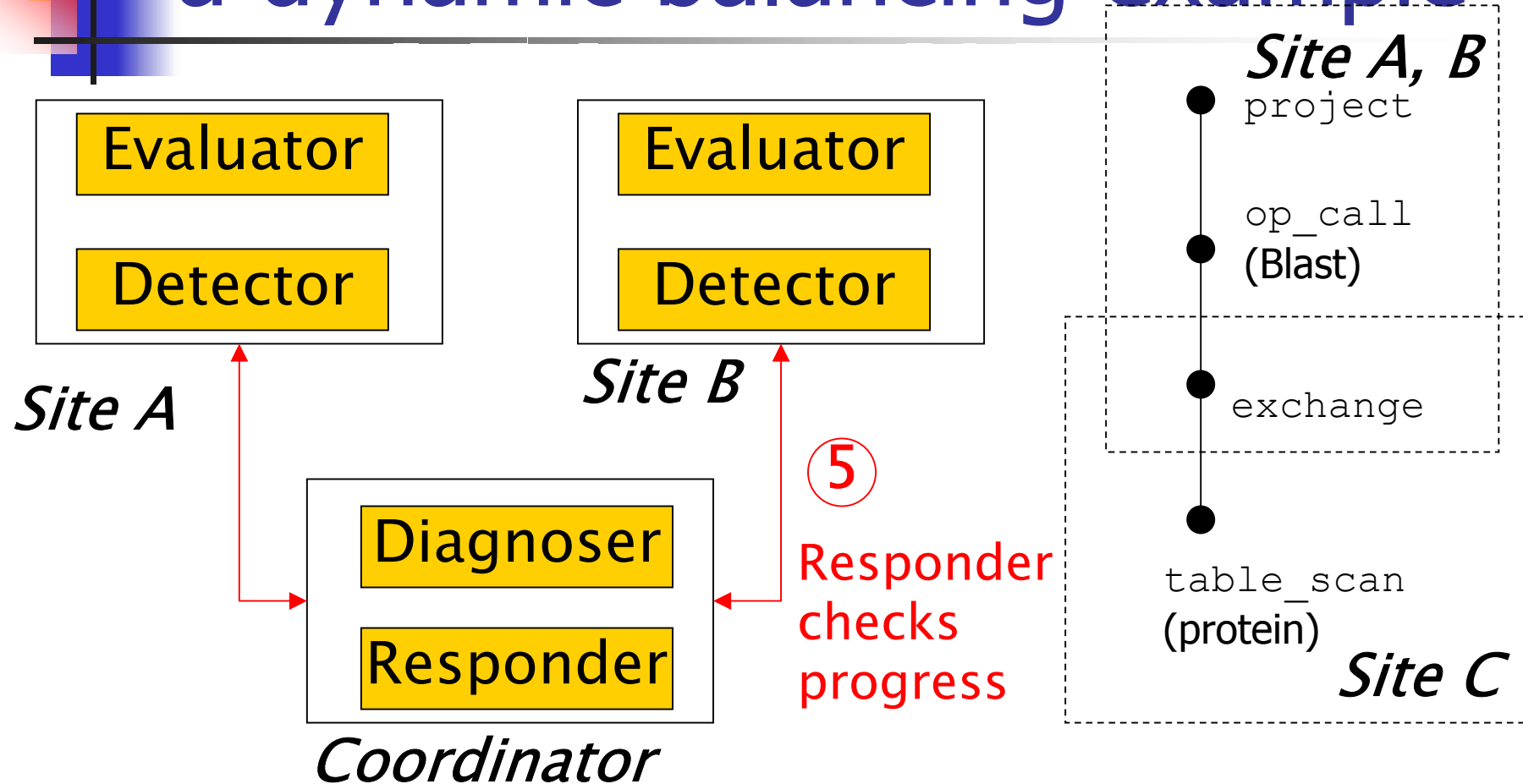
Coordinator

④

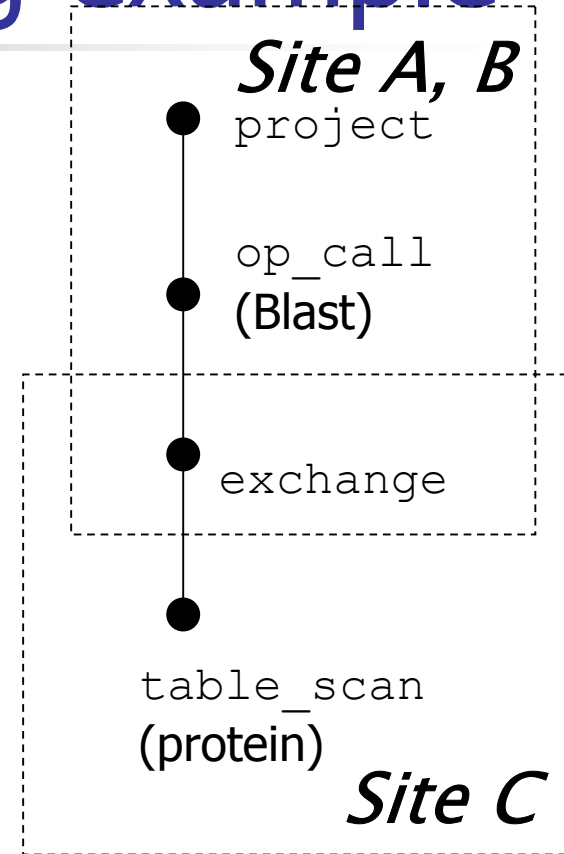
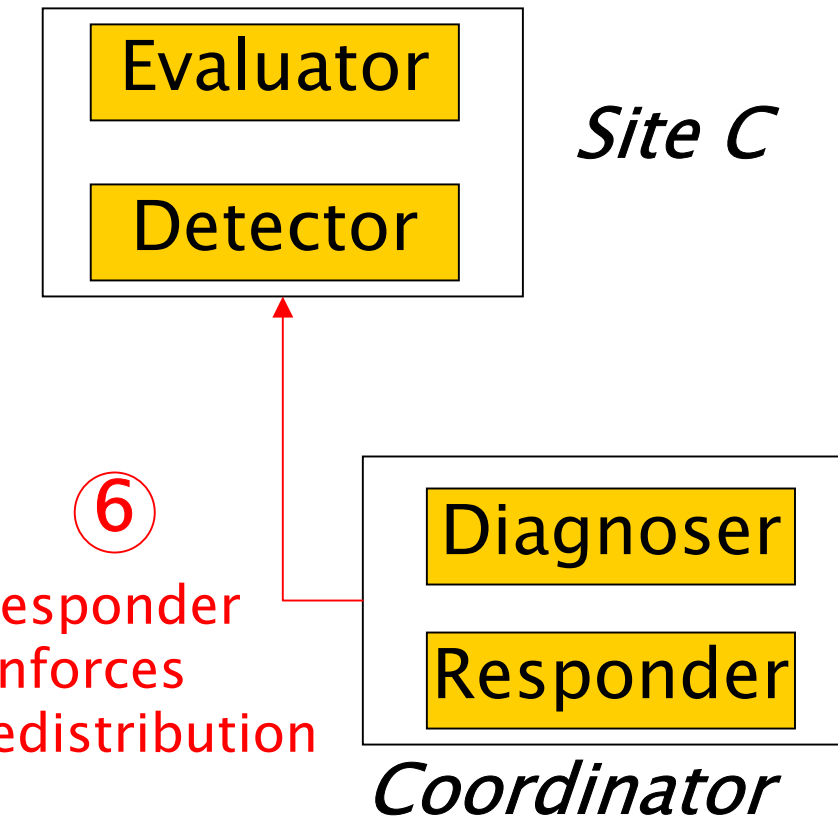
Diagnoser
notifies
responder



Instantiating the framework: a dynamic balancing example



Instantiating the framework: a dynamic balancing example





Categories of Monitoring

- **M1**: Notifications on the cost of processing a tuple in a partition (tuple processing cost, selectivity, time waiting for input).
- **M2**: Notifications about communication costs (cost of sending a buffer, recipient of buffer, size of buffer).



Monitoring: Parameters

- **M1** notifications:
 - Events produced by evaluator every 10 tuples;
 - Averages are calculated for last 25 events;
 - Threshold for sending notification is a 20% change in cost per tuple.
- **M2** notifications are for every buffer.



Assessment

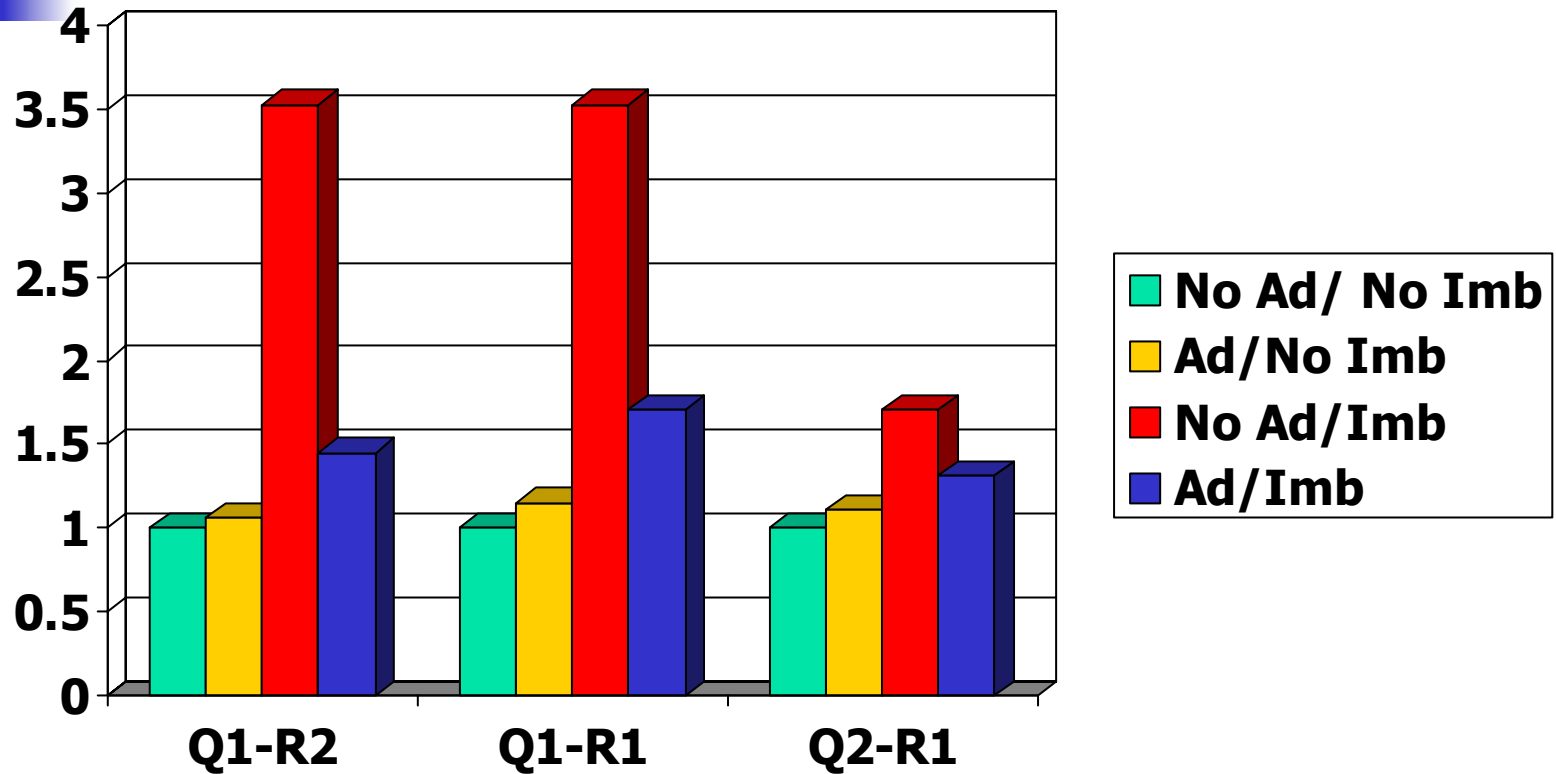
- The assessment component seeks to identify where there is a workload imbalance.
 - Existing Balance Vector $W = (w_1, \dots, w_n)$
 - Proposed Balance Vector $W' = (w'_1, \dots, w'_n)$
- Notify response component if:
 - $(|w_i - w'_i|) / w_i > \text{threshold (20\%)}$ for some i .
- **A1** assessment considers only **M1** notifications.
- **A2** assessment considers both **M1** and **M2** notifications.



Response

- **If** evaluation not close to completion and If progress since last adaptation $> x\%$
then apply new distribution vector W'
- **R1** response is *retrospective*: resends tuples to consumer based on W' .
- **R2** response is *prospective*: no tuples are resent, but future tuples are sent to consumers based on W' .
- Note that **R1** can be applied in more cases than **R2**.

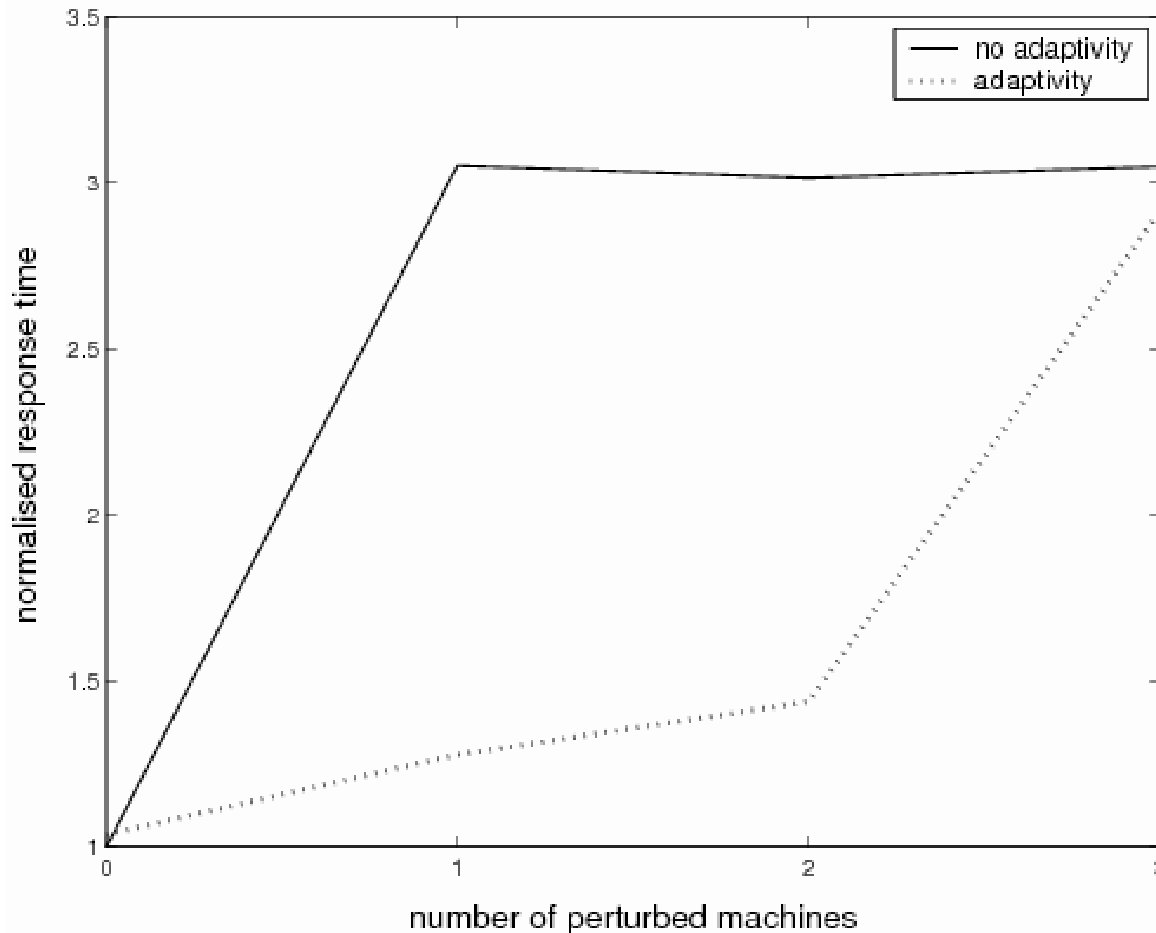
Normalised Results



Q1: call a remote WS, **Q2:** join two remote databases

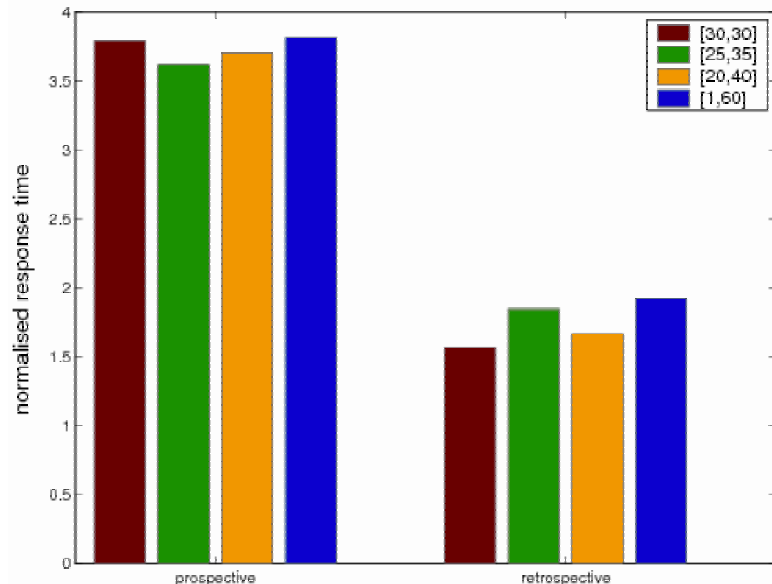
R1: retrospective adaptations, **R2:** prospective adaptations

Changing the number of perturbed machines



Varying Imbalance Dynamically

- Q1: M1 monitoring, A1 assessment.
- Normally distributed perturbation, in range $[30,30]$, $[25,35]$, $[20,40]$, $[1,60]$.
- Lessons:
 - Retrospective generally better at this level of perturbation.
 - Neither prospective nor retrospective thrown by varying imbalance.





Lessons Learnt

- AQP on the Grid can be based upon asynchronous communication of loosely coupled components, which support a pub/sub interface.
- Experiments with imbalances show that the approach can yield significant performance improvements in practice.
- It is relatively easy to tune and calibrate the monitoring-assessment-response policies.



Future Work

- More adaptations
 - Adapting to changes in the resource pool
 - Adapting to changing network bandwidth
- Better adaptations
- Experimentation in a more Grid-like environment (e.g., PlanetLab)



where to find out more

Polar*

Grid-enabled Adaptive Query Processing

www.ncl.ac.uk/polarstar

Complete list of publications

www.cs.man.ac.uk/~gounaris

OGSA-DQP

Grid middleware to query distributed data sources

www.ogsadai.org.uk/dqp