Managed by

Online Block Size Adaptation for Enhanced Transmission of Large Data Volumes in OGSA-DAI

CoreGRID Workshop on

Grid Programming Model
Grid and P2P Systems Architecture
Grid Systems, Tools and Environments

Crete, June 2007

Participants

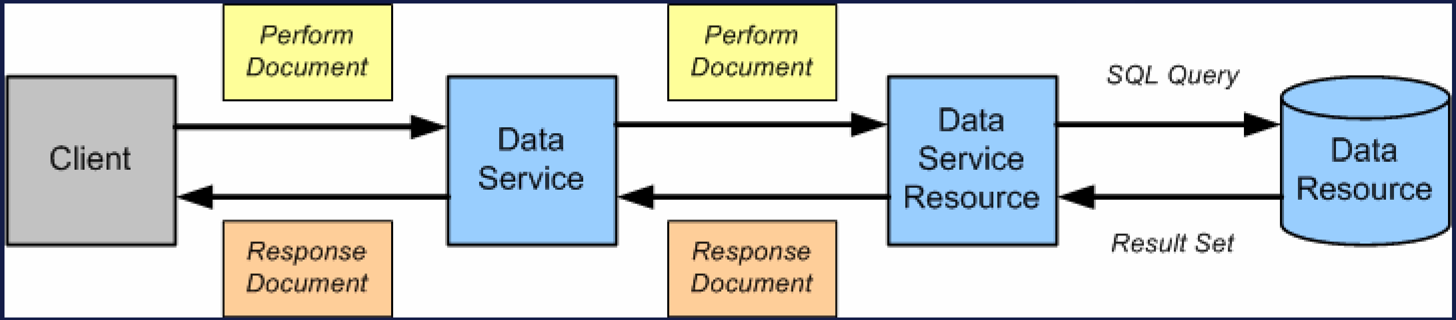
- University of Cyprus (Marios Dikaiakos, Anastasios Gounaris)
- University of Manchester (Rizos Sakellariou)
- Technological Institute of Thessaloniki (Christos Yfoulis)

Talk Outline

1. Problem description
2. Solution
3. Experimental results
4. Conclusions – Future Work

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.



Local experiments - Set up Info

Data:

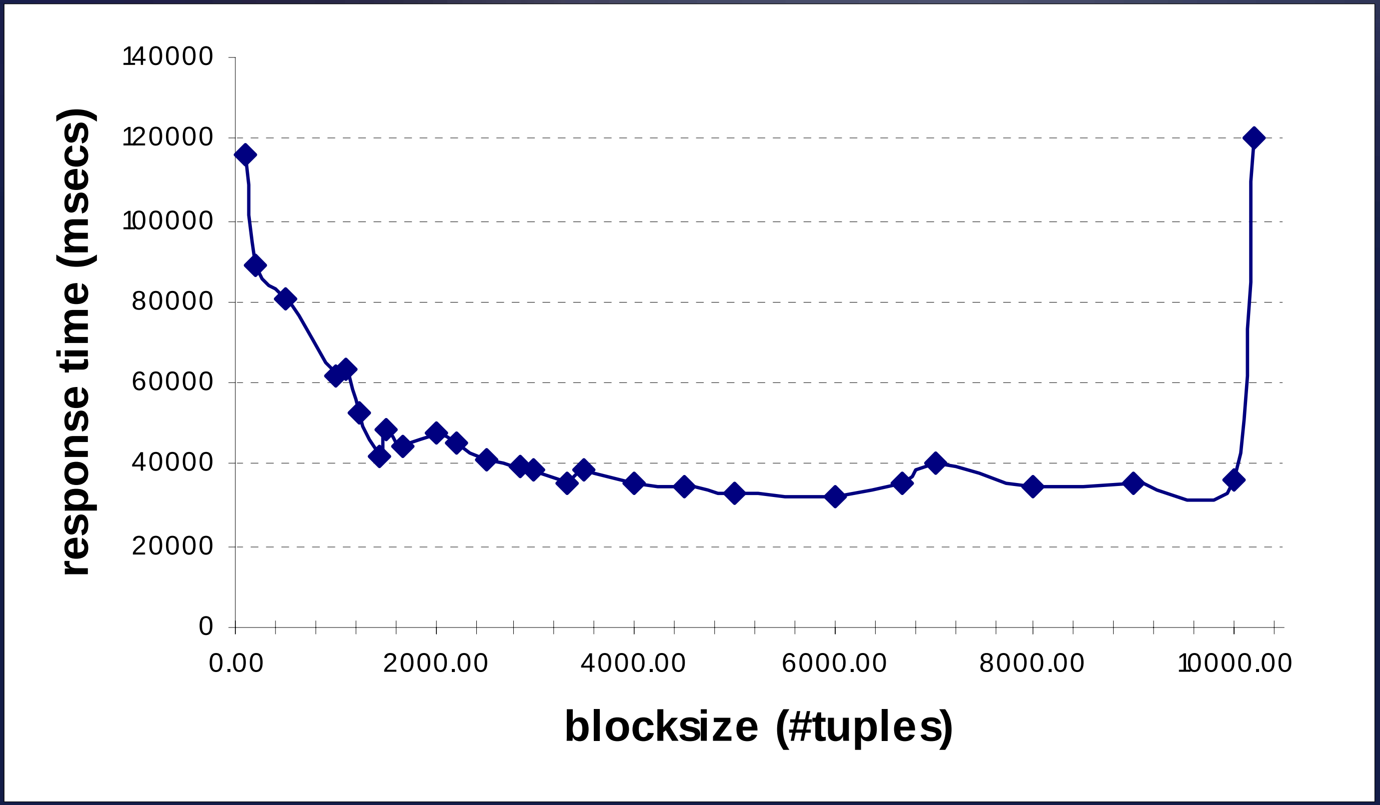
- 100K tuples
- 100bytes average tuple size
- Overall, 10MB of raw data + XML overhead shipped from a WS to a client
- Stored in MySQL

Task

- Retrieve data from store through an OGSA/DAI service
- Send data to client in blocks asynchronously (pull mode) (*)

(*): OGSA-DQP default block size (within Table scan) is 1000 tuples

Overall results

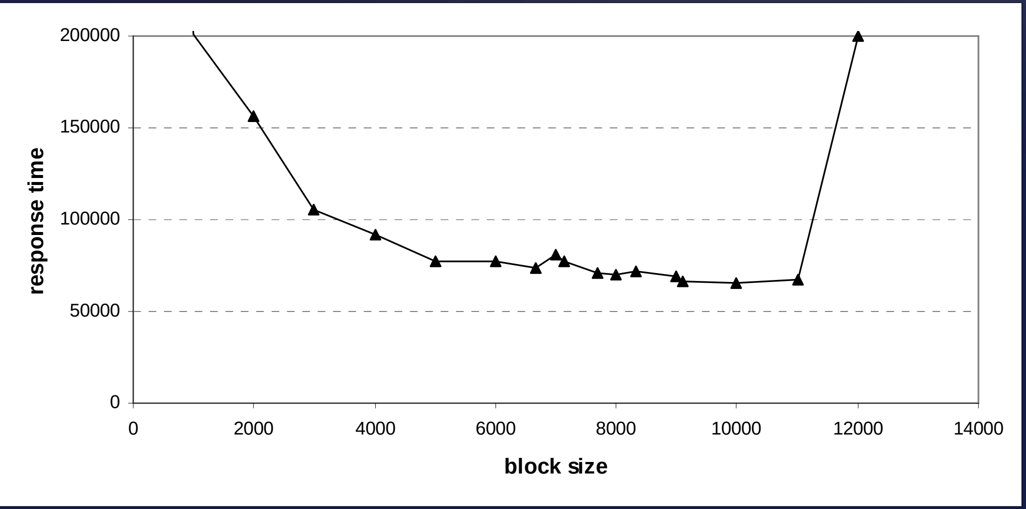


With block sizes slightly larger than 10K tuples, we encounter memory shortage problems

Remote experiments

Server machine at UoM (rpc248)

Client machine connected to UCY network



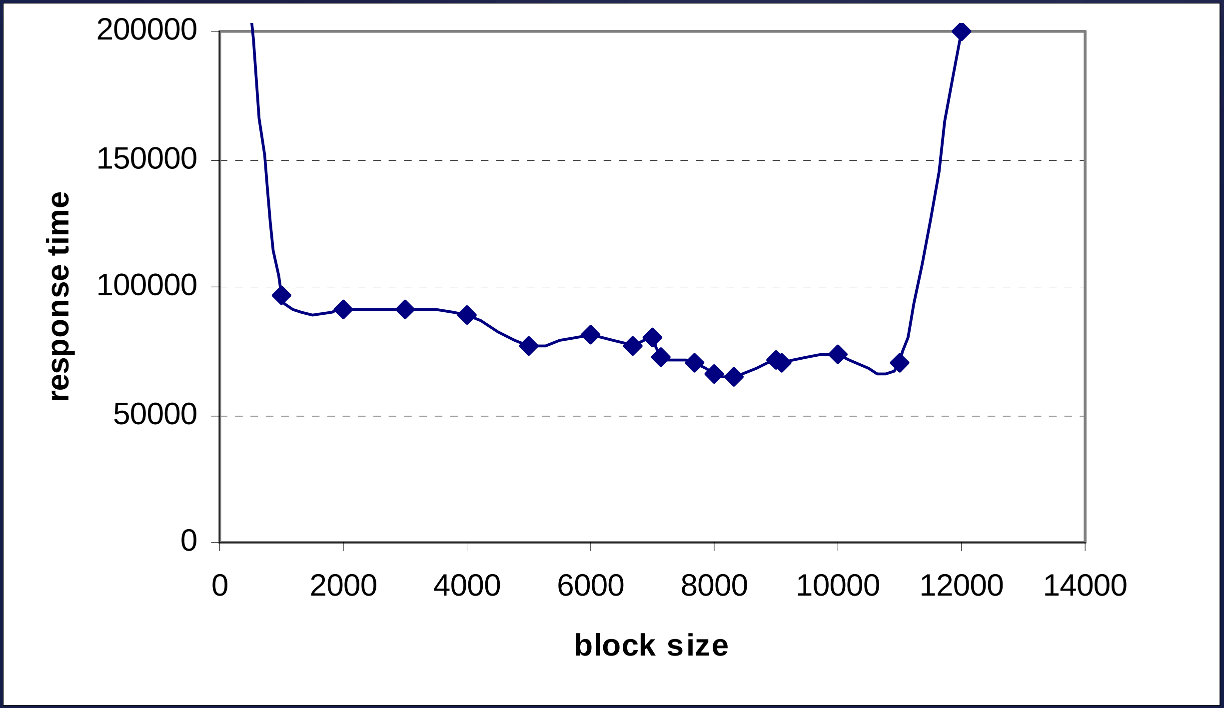
Compared with previous slide, optimal block size is 10K instead of 6K.

Performance degradation:

- **when block size is 6K (local profiling): 17%**
- **when block size is 1K (default OGSA-DQP): > 300%**

Another setting

Client and server are connected through a rather unstable wireless connection + 100Mps LAN



Now, the optimal size has changed again (8K approx.).

Main observations

The optimal block size depends on

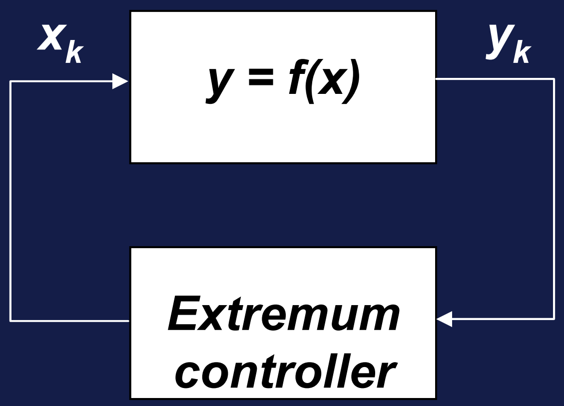
- **The actual connection**
- **The data itself (avg. tuple size)**

Main challenges:

- No model
- Significant noise
  - Local minima
- Requirement for fast convergence

On extremum control

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.**



Switching extremum control

Let *y* be the performance metric and x_k the block size at the *k*th step. Then

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

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 lead to performance improvement, or a decrease has lead to performance degradation.
- Otherwise, the block size must become smaller.

Extensions

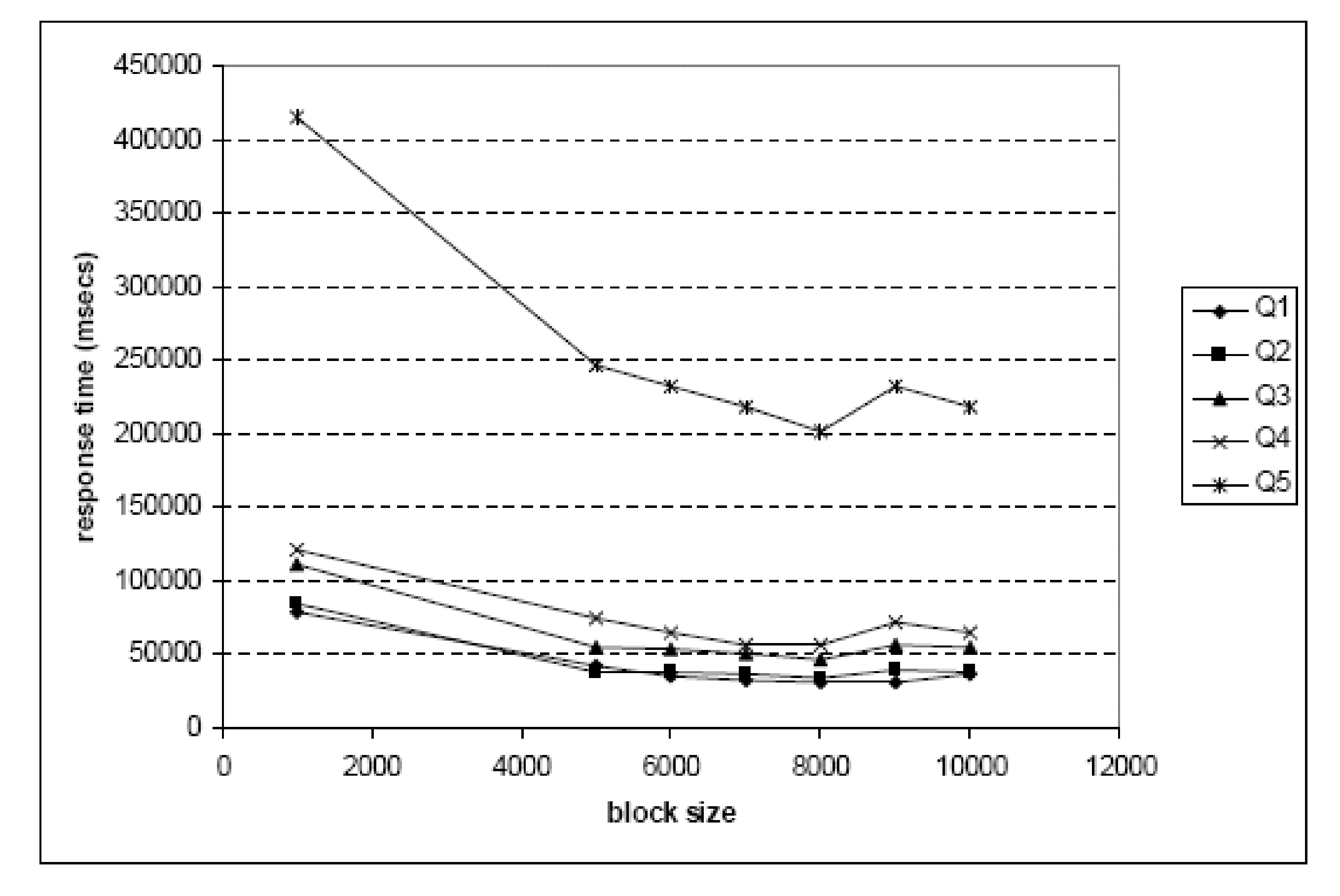
- Linear models are not suitable, thus it is better the *gain* to be adaptive

$$gain = b * \Delta x * \Delta y / y$$

- The impact of noise must be mitigated. To this end, the values of *x,y* are the average over a window
- To enable continuous search of the block size space, add a dither signal

Evaluation

- We used 5 queries over data from the TPC-H benchmark



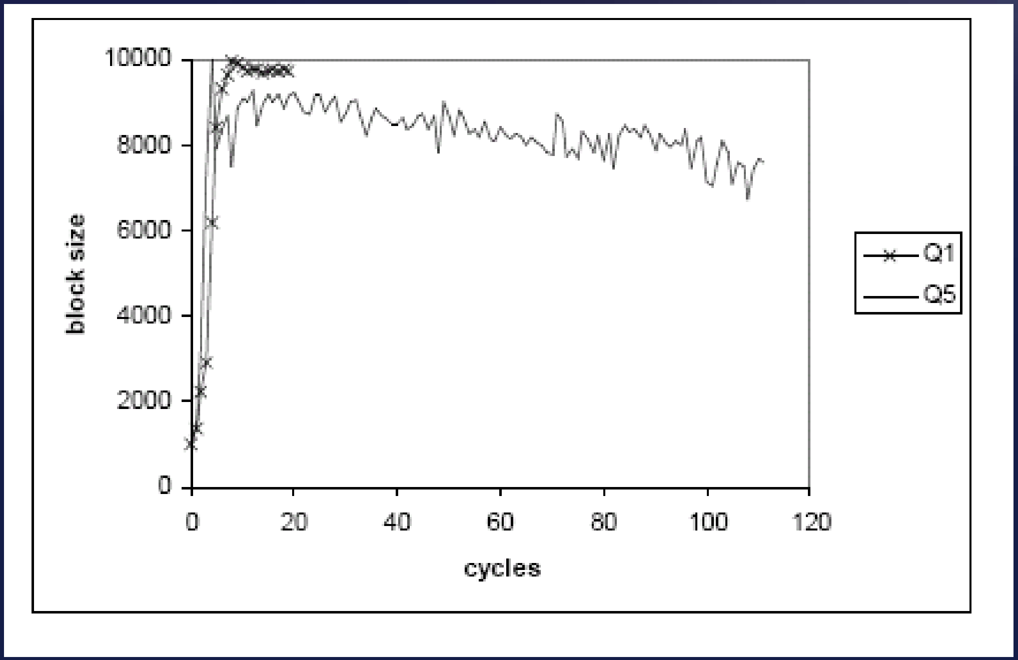
Results

Starting point: 1000 tuples, b = 15

query	*dynamic policy*	*static policy (block size fixed at 1000 tuples)*
Q1	1.152	2.581
Q2	1.191	2.557
Q3	1.192	2.404
Q4	1.154	2.178
Q5	1.02	2.06

The performance degradation drops by an order of magnitude or less

Intra-query behavior



- The technique is characterized by ***stability*** and ***quick convergence***
- However, ***overshooting*** is avoided only by imposing hard upper and lower limits.
- Also, the oscillation in individual runs are rather significant

Conclusions

Techniques from control theory (such as extremum control), seem very promising when a performance metric must be optimized on the fly in these conditions

Future work:

- Investigate model-based techniques
- Investigate more advanced techniques that address the afore-mentioned limitations