# QoS in Grid Computing


**Daniel A. Menascé** • *George Mason University*
**Emiliano Casalicchio** • *University of Rome "Tor Vergata"*

Grid computing is already a mainstream paradigm for resource-intensive scientific applications, but it also promises to become the future model for enterprise applications. The grid enables resource sharing and dynamic allocation of computational resources, thus increasing access to distributed data, promoting operational flexibility and collaboration, and allowing service providers to scale efficiently to meet variable demands.[1,2]

Large-scale grids are complex systems composed of thousands of components from disjoined domains. Planning the capacity to guarantee quality of service (QoS) in such environments is a challenge because global service-level agreements (SLAs) depend on local SLAs.[3] In this column, we provide a motivating example for grid computing in an enterprise environment and then discuss the how resource allocation affects SLAs.

## The Enterprise Grid

For our example, let's consider a large insurance company (IC) that offers coverage for vehicles, boats, homes, and businesses. The IC's primary goal is to increase its profit by minimizing risks and attracting or retaining more customers. In the insurance business, the premiums paid by customers are a function of the risk posed by the insurance policy. Traditionally, ICs use a combination of actuarial data with some minimal personal data to calculate the risk associated with a policy and, thus, its premium. If the assessment yields an excessive risk, premiums increase, which means the IC could lose customers.

In an effort to increase its profits, our IC is moving toward a highly customized risk-assessment model (RAM). Under this approach, the customized RAM queries a much larger group of information sources about a customer to get a richer set of inputs. Our IC's previous model uses large categories (such as "all non-smoking straight-A male college students under the age of 25"), but the new model provides a risk assessment specific to a given customer (besides being a "non-smoking straight-A male college student under the age of 25," John Doe is 23, a student in the Computer Science Department at George Mason University, a member of the IEEE, and part of a programming team that won a regional prize in an ACM-sponsored programming contest. He also has a very good credit record, undergoes a physical exam every year and is in perfect health, and has a clean record with federal, state, and local law-enforcement agencies). Clearly, customized RAMs are much more sophisticated and significantly more compute- and data-intensive. Such models are decomposed into many parallel tasks that run at various grid nodes.

The IC plans to establish a Web portal through which prospective customers can request three types of insurance-policy quotes:

- Immediate requests use simple RAMs and can return results in a few seconds.
- Non-immediate requests use more complex RAMs, but can still return results in a few minutes.
- Delayed requests use fairly sophisticated RAMs and can take hours to process.

In the latter case, users receive an email with a link to the portal, where they can view details about their quotes. Thus, customers might get potentially lower premiums if they're willing to wait longer for the more sophisticated and complex RAMs to execute.

The IC doesn't want to invest additional computing resources for running new models, so it uses a grid to harness the unused cycles of all the computers connected to its worldwide network — from desktops to mainframes. The grid that supports the IC risk-assessment application will

 1089-7801/04/$20.00 © 2004 IEEE Published by the IEEE Computer Society 

**Table 1. Parameters for grid computing resource allocation.**

| Resource | Speed (millions of cycles/sec) | Cost (dollars/sec) |
|---|---|---|
| 1 | 1,000 | 0.10 |
| 2 | 2,000 | 0.25 |
| 3 | 3,000 | 0.60 |

schedule resources according to RAM type and the computational requirements needed to evaluate the three request categories. The new customized RAMs draw their inputs from many data sources, including health-insurance companies, law-enforcement agencies, financial organizations, departments of motor vehicles, professional organizations, and various levels of government agencies. They can also use weather-related data sources to assess risk for home and business insurance policies.

## Resource Allocation

SLAs are contracts between service providers and users, specifying acceptable QoS metric levels; they usually associate a cost (and sometimes penalties for noncompliance) with a desired level of service. An important challenge in grid environments is how to monitor and enforce SLAs when many users share the same resources, especially because a key part of a grid environment's definition is that it provide nontrivial QoS.[1]

Application-level SLAs must be mapped to resource-level SLAs. When negotiating SLAs with lower-level resources, the grid's global resource scheduler must realize that different service providers might offer the service at different levels for different costs. Thus, an interesting optimization problem in a grid environment is how to select services and service providers in a way that achieves the global SLA with minimum cost.

Going back to our IC example, let's say that a RAM evaluation requires $NC$ million CPU cycles, must be finished in at most $T_{max}$ time units, and that the computation cost must not exceed $C_{max}$ dollars. Assume that at most three compute resources can be allocated for the evaluation, and let $s_i$ ($i$ = 1, 2, 3) be the speed of resource $i$ in millions of cycles/sec and $c_i$ ($i$ = 1, 2, 3) be the cost in dollars/second to use resource $i$. We'll break down the evaluation in up to three independent parallel tasks. Thus, the three compute resources have seven possible allocations: (1), (2), (3), (1,2), (1,3), (2,3), and (1,2,3).

Let $T$ and $C$ be the execution time and cost of a given allocation, respectively. Some allocations might not be feasible in terms of satisfying the global SLA $T_{max}$ or the global maximum cost $C_{max}$. If only resource $i$ is used, then

$$T = \frac{NC}{s_i} \le T_{max} \quad i = 1, 2, 3, \tag{1}$$

$$C = c_i \times \frac{NC}{s_i} \le C_{max} \quad i = 1, 2, 3. \tag{2}$$

Equation 1 says that the total execution time at resource $i$ can't exceed the global SLA $T_{max}$. Equation 2 is the cost constraint. The time and cost constraints when two resources $i$ and $j$ are used are given by

$$T = \max\left\{\frac{NC_i}{s_i}, \frac{NC_j}{s_j}\right\} \le T_{max}, \tag{3}$$

$$C = c_i \times \frac{NC_i}{s_i} + c_j \times \frac{NC_j}{s_j} \le C_r, \tag{4}$$

$$NC_i + NC_j = NC. \tag{5}$$

If all three resources are used, then

$$T = max\left\{\frac{NC_1}{s_1}, \frac{NC_2}{s_2}, \frac{NC_3}{s_3}\right\} \le T_{max}, \tag{6}$$

$$C = \sum_{i=1}^{3} c_i \times \frac{NC_i}{s_i} \le C_{max}, \tag{7}$$

$$\sum_{i=1}^{3} NC_i = NC. \tag{8}$$

If we ignore the cost constraint, the solution that minimizes RAM-execution time is the one that allocates cycles proportionally to a resource's speed. Thus,

$$N_i = NC \times \frac{s_i}{\sum_{j=1}^{3} s_j}. \tag{9}$$

Let's look at the general case in which we take into account both maximum execution time and maximum cost. One version of the optimization problem to be solved by the grid's resource-allocation mechanism is, "Find the feasible allocation that satisfies the performance constraint $T \le T_{max}$ at minimum cost." A dual problem is, "Find a feasible allocation that minimizes execution time $T$ and satisfies the cost constraint $C \le C_{max}$."

Consider the following numerical example: $NC = 10^7$ million cycles, $T_{max}$ = 4,800 sec, $C_{max}$ = US$1,500. Table 1 gives the speed and cost values for the three compute resources.

For each possible resource allocation, we solve the optimization problem of finding the allocation of cycles that minimizes the execution time $T$, while satisfying the cost constraint $C \le C_{max}$. Table 2 shows the solutions. Allocations marked in italics are not feasible (meaning they violate the execution time SLA or the cost constraint); allocations marked in bold are feasible. The last row corresponds to the optimal allocation, which executes the RAM in 2,593 seconds at a cost of $1,352.

The first two allocations are infeasible because they violate the execution time constraint (a maximum of 4,800 seconds). The third and fifth allocations are infeasible because they violate the cost constraint (a maximum cost of $1,500). Interestingly, in some allocations (such as the last one),

**Table 2. Solutions to the resource-allocation problem.***

| Allocation | Cycle allocation ($NC_i$) in millions of cycles | | | Execution time (sec) | Cost ($) |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | | |
| *(1)* | *10,000,000* | — | — | *10,000* | *1,000* |
| *(2)* | — | *10,000,000* | — | *5,000* | *1,250* |
| *(3)* | — | — | *10,000,000* | *3,333* | *2,000* |
| **(1, 2)** | **3,333,333** | **6,666,667** | **—** | **3,333** | **1,167** |
| *(1, 3)* | *4,800,000* | — | *5,200,000* | *4,800* | *1,520* |
| **(2, 3)** | **—** | **6,666,667** | **3,333,333** | **3,333** | **1,500** |
| **(1, 2, 3)** | **2,592,593** | **5,185,185** | **2,222,222** | **2,593** | **1,352** |

*Allocations in bold are feasible; italics are infeasible.

the faster and more expensive node — node 3 in this case — is used only during part of the execution time because of the cost constraint. In fact, because 2,222,222 million cycles are allocated to it and its speed is 3,000 million cycles/second, this node is used for 741 seconds, or 28.6 percent of the execution time.

## Final Remarks

If we have $n$ resources, and if a computation can be scheduled into any number of resources from 1 to $n$, the total number of possible allocations to examine grows in a combinatorial way:

$$\sum_{k=1}^{n}\binom{n}{k} - 1 = 2^n - 1. \quad (10)$$

Thus, it might be computationally expensive to analyze all possible allocations and solve an optimization problem for each one. In this case, the grid's resource scheduler might have to use heuristics to obtain close to optimal solutions in a more efficient manner.

A real environment is more complex than the example we've considered here: the computation might have a structure that reflects interdependencies among tasks, and we'd have to consider communication delays for data transfers and coordination overheads. Nevertheless, the example discussed here highlights two important issues that arise when scheduling computations into the nodes of a grid — namely, SLA compliance and cost constraints.


### References

1. I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, 2nd ed., Morgan Kaufmann, 2004.
2. F. Douglis and I. Foster, "The Grid Grows Up," *IEEE Internet Computing*, vol. 7, no. 4, 2003, pp. 24–26.
3. A. Leff, J.T. Rayfield, and D.M. Dias, "Service-Level Agreements and Commercial Grids," *IEEE Internet Computing*, vol. 7, no. 4, 2003, pp. 44–50.



### Acknowledgments

Daniel A. Menascé's work is partially supported by grant NMA501-03-1-2022 from the US National Geospatial-Intelligence Agency.



**Daniel A. Menascé** is a professor of computer science, codirector of the E-Center for E-Business, and director of the MS in E-Commerce program at George Mason University. He received a PhD in computer science from UCLA. Menascé is author of the books *Performance by Design*, *Capacity Planning for Web Services*, and *Scaling for E-Business* (Prentice Hall, 2004, 2002, and 2000). He is a fellow of the ACM and a recipient of the 2001 A. A. Michelson Award from the Computer Measurement Group. Contact him at menasce@cs.gmu.edu.

**Emiliano Casalicchio** is a research assistant professor at the University of Rome "Tor Vergata," Italy, from which he received a PhD in computer engineering. His research interests include performance modeling, analysis and simulation, networked computer systems, and Web-based architectures. Contact him at casallicchio@ing.uniroma2.it.