Centralized vs. Decentralized Design in Internet-Scale Applications
Statement of Interests

Adriana Iamnitchi
Department of Computer Science
The University of Chicago
anda@cs.uchicago.edu

My research interests are in the area of distributed computing. My first contact with the centralized vs. decentralized design conflict occurred while working on a branch-and-bound algorithm for Internet connected resources. Given the large number of resources needed to solve very large problems, and given the unreliable target system, the obvious choice seemed to be a fully decentralized approach. However, the existent (and successful to some extent, in that that they solve problems using hundreds of processors simultaneously) branch-and-bound algorithms use mostly the master-worker design, hence they are centralized. The centralized approach is unscalable for a very large number of resources, e.g., tens of thousands of worker processes. We proposed a fully decentralized branch-and-bound algorithm. We tested our algorithm in a simulation environment and we obtained very encouraging results[1] regarding reliability, scalability, and overall performance. However, until this algorithm is implemented, the existent master-worker branch-and-bound implementations will solve some of the scientific problems.

The example above is from the class of applications that would benefit from the large collection of resources existent in Internet. Another class of problems that involve scalability issues is the middleware services—protocols and software that support the parallel execution of Internet-scale applications. I am currently interested in resource discovery mechanisms on Internet-scale collections of resources. My current research is in the context of Globus[2] project.

Because of its simplicity, a centralized design was initially chosen for implementing the Information Service. This centralized solution was conceptually simple: a new entity that joins the testbed registers itself to the central Information Service server. When a request for information is addressed to the server, the server has quick access to all existent information and is able to answer efficiently. After the centralized design became a bottleneck and a reason for multiple failures with the growing number of resources in the testbed, the decen-

[1] Adriana Iamnitchi and Ian Foster. A Problem-Specific Fault-Tolerant Mechanism for Asynchronous Systems. To appear in *Proceedings of the International Conference on Parallel Processing*, 2000.

[2] http://www.globus.org

tralized design became obviously a better option. In a decentralized Information Service, there is no central point of control. Therefore, challenges like providing accurate information in short time and implementing security policies in a highly heterogeneous environment become more difficult to overcome. In addition, new problems are to be solved, like providing a flexible frame while assuring that communication between various Information Servers can still be possible.

Scalability and reliability are the key requirements for most Internet-scale applications. To achieve scalability and reliability, I would argue that computationally expensive applications *need* at least a partially decentralized design. Some applications that are not very computationally intensive (like the examples from the TWIST 2000 web page: Amazon.com, eBAY and AOL) can benefit from the simplicity of the centralized design.

I find the topic of TWIST 2000 very interesting and challenging. I am sure that my understanding of the various scalability issues would be enriched by the participation to this workshop.