<u>Ways to improve the infrastructure</u>

1. Shared Input object

Each application contains an Input object that are needed by all the tasks for performing computations. For instance, in TSP, each computational task needs the cities that are to be explored and which represent the points in the 2D Euclidean plane. In the existing infrastructure, that object is sent to each newly constructed task and thus involves a lot of overhead. This can be avoided by having a globally accessible read-only input object that all the tasks specific to a particular application can use during execution.

2. Client waiting time

In the existing infrastructure, the client submits the task to Space and waits until the result is available. i.e. it waits for the entire period of job execution. A lot of network bandwidth is wasted in the process and it also prevents the client from doing any other useful work. To avoid this, the client can register a callback method with Space that the latter uses to communicate with the client when the final result is computed. As soon as the client puts the root task in Space and register the callback method, it can return.

3. Limitations with respect to Space

The Compute Space acts as a store of Task objects and has a number of Computer Servers taking tasks from it and executing them. The current infrastructure has only one Compute Space making it a single point of failure. As in the JICOS architecture, two or more Compute Spaces can be installed to eliminate this. Only one Compute Space needs to be active at any time and the other Space(s) can be in passive mode (similar to the configuration of modern-day firewalls); the active Space can periodically synchronize its content (the data in the task queues, the information about the registered computers, etc.) with all the passive Spaces. If the active Space goes down, one of the passive Spaces can take over. The Computers can periodically send heartbeat signals to the active Space to determine its health and if that Space goes down, they can obtain the remote reference to the new Compute Space that becomes active.

Another aspect that is worth discussing is the issue of multi-threading in Space. In the present architecture, the Space creates a new thread for each Computer that registers with it. This poses a limitation on the number of Computers that can connect to the Space. Also, as the number of computers increases, with all the enhancement features turned on, the workers in the Computers will start sending tasks at a speed that may be beyond the processing capacity of the Space, which in turn would affect the performance of the application. This again can be mitigated by having a network of Spaces with each Space taking control over a given number of Computers.