AMELIORATING COMMUNICATION LATENCY & EXPLOITING MULTI-CORE HOSTS

API Design

Figure 1 shows the design of the API and explains how the same framework can support multiple combinatorial optimization problems (eg. TSP, Integer Programming, etc.) as well as other applications like Mandelbrot Set visualization and Fibonacci Sum.

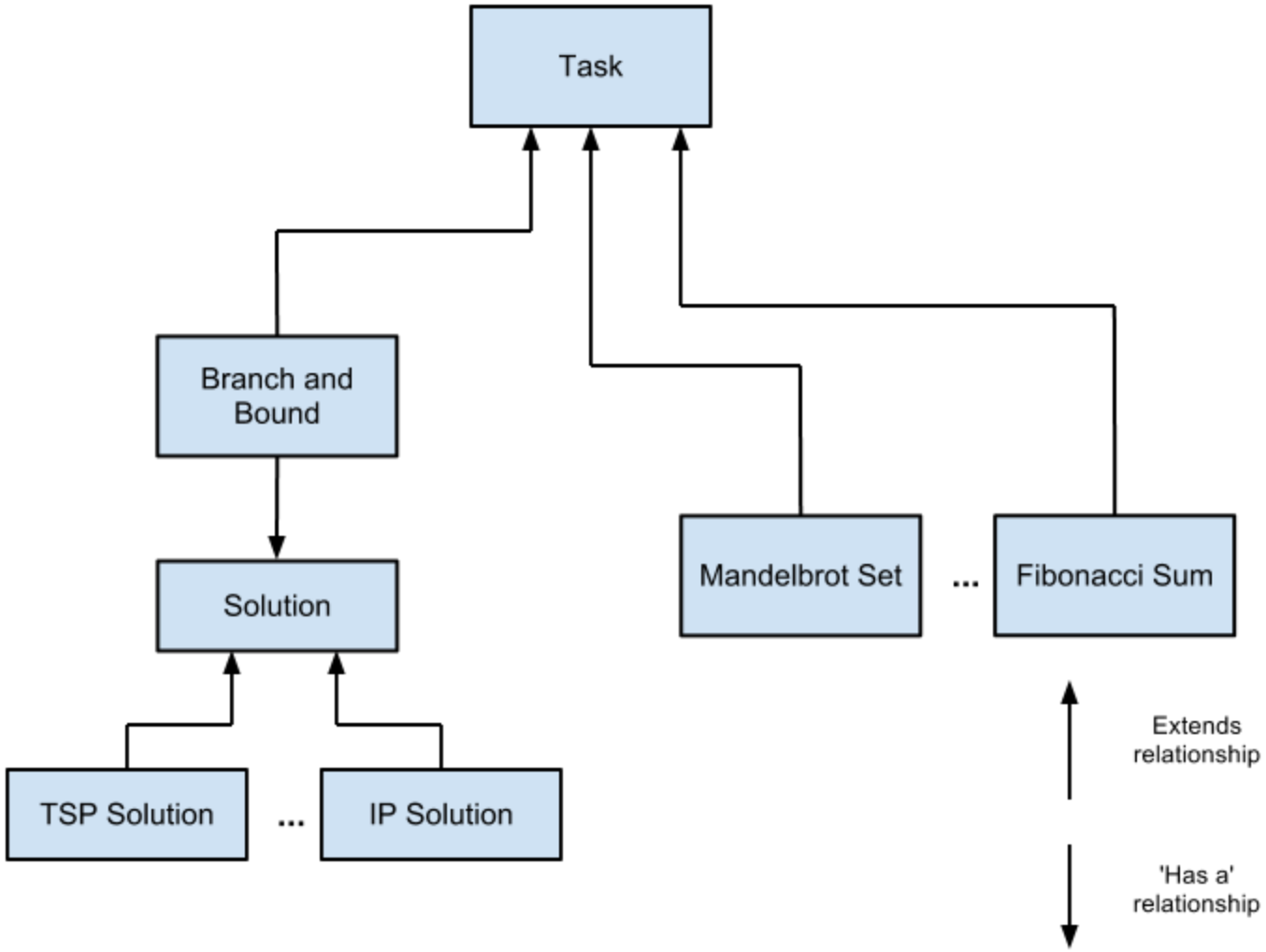

Fig.1 API Design

Infrastructural Enhancements

In order to improve the performance of the applications, several enhancements such as reducing communication latency and overlapping computation with communication have been done.

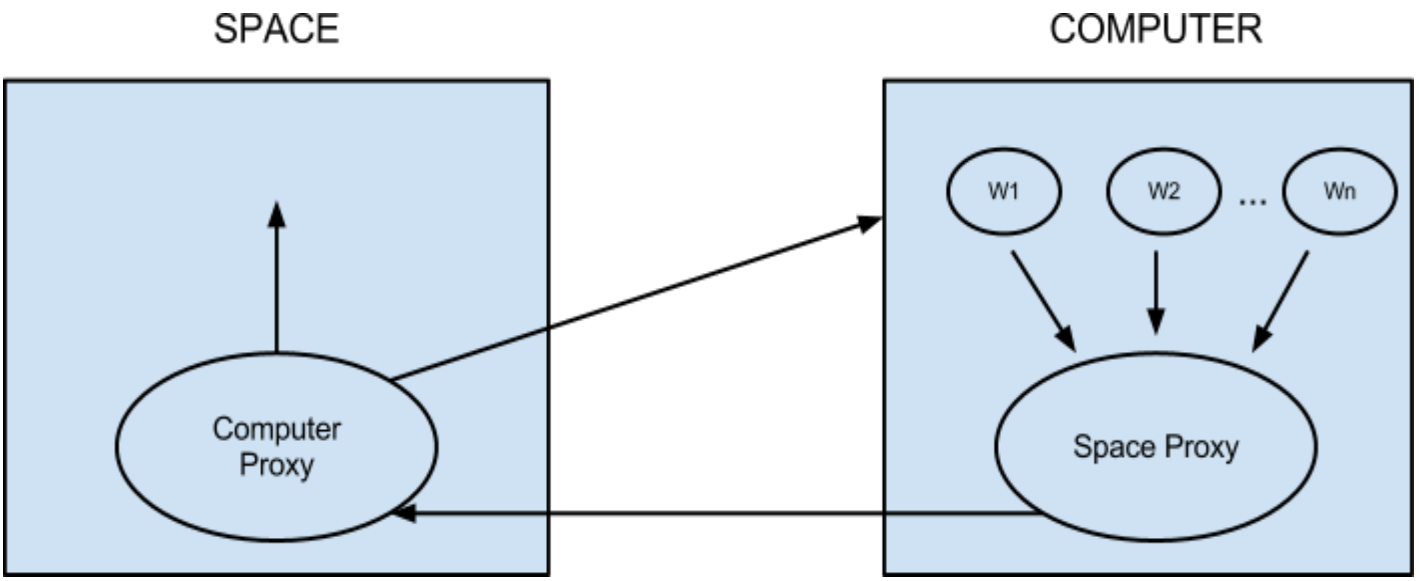

Fig. 2. Space - Computer Interaction

The Space - Computer interaction is as follows:

1. When a Computer registers with the Space, an RMI reference to Computer Proxy (which implements Computer2Space) is sent back to the Computer.
2. The Computer stores this RMI reference as the Space Proxy. The Computer communicates with the Space only through this Space Proxy.
3. When a task is added to the ready queue in the Space, the Computer Proxy takes it and calls the Computer's execute method.
4. If the amelioration enhancement is turned on, the Computer adds this task to its task queue (a blocking queue) and the RMI call returns. The worker thread(s) that are waiting on this task queue take the task, execute it and puts the result back into Space via the Space proxy (a remote reference to the Computer Proxy). This process is asynchronous so as to overlap computation with communication. The number of worker threads is equal to the number of processors that are available to the Java virtual machine on which the Computer is running. If this particular enhancement is turned off, there will be only one worker thread irrespective of the number of processors.
5. If the amelioration enhancement is turned off, i.e. if the communication doesn't overlap with the computation, the Computer Proxy blocks on the execute call and returns only when the Worker thread(s) send the result back.

Fault Tolerance

The following enhancements have been made in order to accommodate faulty Computers. The Computer Proxy maintains a list of all the tasks that have been sent to the Computer and whose results have not been received . Before calling the Computer's execute method, the Computer Proxy adds that task to the list. When the task is executed on the Computer and the result returned back to the Computer Proxy, it is removed from the list. In case, the Computer crashes, the tasks that are left behind in that list (tasks that have been sent to the Computer and whose results have not been sent back) are added to the ready queue of the Space so that the other Computers can execute

them.

Space-runnable Tasks

When a task's encapsulated computation is little less complex than reading its inputs, it is faster for the Space to execute the task itself than to send it to a Computer for execution. This is because the time to marshal and unmarshal the input plus the time to marshal and unmarshal the result (added with the network latency) is more than the time to simply compute the result. This is achieved by defining a method called executeOnSpace() in Task that returns a boolean value indicating whether or not the task can be executed on Space. The default behavior is to make all the composition tasks execute on Space. This can be overridden by making the application specific task classes (e.g. a FibonacciTask) override this method and defining an alternate behavior.

Enhancement Features

The infrastructure has been built in such a way that the enhancement features mentioned can be turned on or off. This is achieved by storing the enhancement information in a properties file and loading it during execution time.