**Server Engineer needed**

We need an experienced server engineer to design and implement a scalable web service for the Linux platform. We have discussed writing the web service in Java, C++ and even PHP and part of the project will be to evaluate and decide on the appropriate language and technology to use.

On the server side the service will integrate with the Modelica , EnergyPlus and Radiance simulation engines. We have specialists in these applications to assist you. These server side applications receive input through stdin and output via stdout. However we prefer not to integrate this way due to limitations this would impose. Also stdin/stdout pipelines tend to be somewhat fragile. We prefer a more robust integration if possible where the web service makes C++ function calls directly to the Modelica , Energy Plus and Radiance API. The web service will communicate with our client application which is running in a JavaScript enabled web browser.

**Here are some of the attributes that the completed system will have:**

**Stateful** – The system will maintain sessions with users. This will allow users to make multiple requests setting values, making changes and retrieving results without resending the entire dataset.

**Scalable** – The system should scale to support 1,000 concurrent users. As more users come online, the system will be able to spawn more threads to handle them and even more virtual servers to handle the load.

**Robust** – The Web service should be able to monitor client connections, and running simulations. It should be able to detect common failure modes and recover, sending the user error messages, and restarting processes on the server as needed.

**Cloud Compatible** – We will likely to deploy the system on a cloud computing platform like Amazon’s EC2 platform.

**Service Oriented Architecture** – The overall architecture will be service oriented so that clients can connect and be made aware of all available services. The self-documenting characteristic of the SOA architecture will be useful in coordinating our client development.

**Standardized network protocol** - the system will you REST, SOAP, XML-RPC or some other standard protocol to communicate with the remote client.

**Standardized result sets** – The system will return a standardized result set in the XML, JSON or other AMF. Our clients will be using JavaScript to make requests and receive results. We are interested in the potential of AMF (Action Message Format) due to its greater efficient handling binary data. However we need to evaluate the maturity of the JSAMF client library.