Grid agent data-flow execution

Nigel Sim

School of Information Technology/School of Maths and Physical Sciences

James Cook University

The goal of Grid computing is to enable generic, distributed, heterogeneous computing resource to be used together to perform arbitrary computing tasks. This goal can be interpreted in a number of ways, and the functionality can be implemented at a number of levels, with the onus for the “Gridification” being implemented somewhere between the operating system and the end application. An important aspect of implementing a pure Grid system is to be able to intelligently reuse existing software components, with minimum modifications. Further, it should be possible to scale the system from individual machines, all the way to commercial supercomputing facilities, once again with minimum modification. Finally, any system must support a realistic development environment, especially concerning the validation of implemented techniques.

We present an agent based Grid computing platform, which defines the system boundaries and responsibilities in such a way to enable the ready reuse and integration of native code libraries into a single processing system, which can be hosted in a heterogeneous grid of computers. The system's primary interface revolves around the R environment for statistical computing. R is widely used in the bio-informatics community, and it's use allows us to lever off existing development and analysis practises to achieve useful outcomes in terms of re-use, and application speed up.

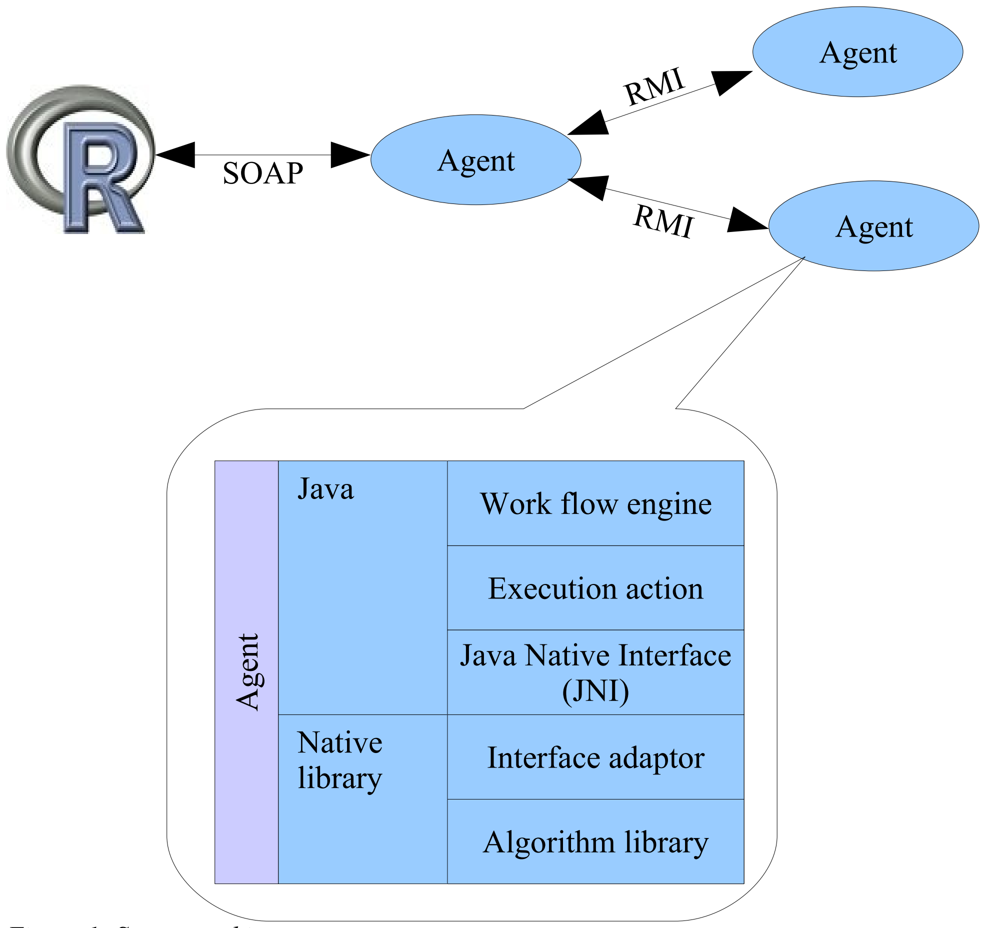

Figure 1: System architecture.

The agents are written in Java for maximum portability. This is essential to ensuring library availability on remote platforms, with Java implementation being packaged and distributed with the main Java archive (JAR). The agents handle all aspects of data handing, including input, output and data type conversion, and pass computations tasks to external Java classes, and native libraries through the Java Native Interface (JNI). This allows the native libraries to only be concerned with computations, and not with the format of the input data. This allows these algorithms to be written in a clean, generic and portable way, and to also have access to a many data sources and data types which may not have been available to the native implementation on its own.

Inter-agent communication is currently handled through RMI/CORBA, allowing for fast communications and data transfer. Administration and workflow control is accessed via a SOAP interface, allowing a wide range of languages to talk to the system, and for the network to be exported as a Grid Service. The system can operate in either an interactive mode, where workflows are actively fed into the system by outside processes, or in a batch mode, where the entire workflow is delivered upfront, making the system suitable of a variety of processing environments.

Execution is represented as a data-flow graph, which is decomposed distributed among the available processing agents. The granularity of the system is automatically adjusted to minimise communications overhead, with data location, relative computational speed, and measured link capacity being investigated to provide workable heuristics for determining job placement.

Future plans include the introduction of a hierarchical scheduling model, which will allow multi-site execution to be performed in an efficient manor by assigning large chunks of the data flow graph to individual sites, and allowing local cliques of agents on site to again decompose the chunk for processing.