A Perspective on the *CoreGRID* Grid Component Model

Françoise Baude

INRIA, I3S-CNRS, University of Nice Sophia-Antipolis,
2004 route des Lucioles, Sophia Antipolis, France
Francoise.Baude@inria.fr

The Grid Component Model is a software component model designed partly in the context of the *CoreGRID* European Network of Excellence, as an extension of the Fractal model, to target the programming of large-scale distributed infrastructures such as computing grids [3]. These distributed memory infrastructures, characterized by high latency, heterogeneity and sharing of resources, suggest the efficient use of several CPUs at once to obtain high performances. To address these characteristics, GCM features

- primitive components that can be deployed on different locations of the grid, through the notion of Virtual Nodes and their associated XML-based deployment descriptors,
- composite and distributed components exporting server and client interfaces of their inner components
- collective interfaces (multicast, gathercast, MxN)
- and, as in Fractal, an open control part giving GCM components full introspection and reconfiguration capabilities.

Moreover, the control part itself can be designed as a composition of distributed GCM components [4], allowing for full expressiveness and more importantly, full adaptability of the control part even at runtime. A GCM membrane (control part) can for instance implement an autonomic adaptation of the parallelism degree of the composite parallel component it controls (a.k.a. behavioral skeleton [1]) i.e. configure the number of inner components working in parallel to achieve a given task.

The GCM specification (API, Architecture Description Language for initially describing a GCM application, and GCMA/GCMD deployment descriptors) has been approved as an ETSI standard by the technical body in charge of grids and clouds.

A reference implementation of GCM relying upon the open source ProActive parallel suite (`proactive.inria.fr`) relies on Virtual Nodes concrete instantiation, and the use of Active Objects to implement distributed components. Server and client interfaces method invocations thus rely on asynchronous method invocations with futures. Moreover, futures are first-class which is key to propagate interface calls, in particular within composite components. Strategies to update future values have also been deeply experimented with, and the way components interact through requests has been formalized and proved correct (using the Isabelle theorem prover) [5]. Behavioural specifications of GCM interfaces enable hierarchical and thus scalable model cheking of whole GCM applications [2].

GCM has been successful in building applications acting as high-level middlewares. MPI-like applications can be executed on top of on any combination of heterogeneous computing resources from different administrative domains i.e. acquired from different

M. Alexander et al. (Eds.): Euro-Par 2011 Workshops, Part I, LNCS 7155, pp. 115–116, 2012.

clusters, grids and even clouds, thanks to a GCM-based substrate [6]. Such substrate handles the efficient and seamless inter domain routing of application-level messages exchanged between processes of the non-embarrassinly parallel application. The obtained performances are competitive in regard to existing implementations of MPI on grids. Handling the routing of application-level messages across domains can be useful in other situations, as when building federations of Enterprise Service Busses [6].

GCM has been given an *SCA (Service Component Architecture)* personality, meaning one can design a GCM application as an SCA one, including the use of SCA intents. Moreover, GCM primitive components can be implemented as BPEL documents (according to the SCA-BPEL specification). The links to orchestration engines through specific GCM controllers allow partner links dynamic adaptation and opens the way for distributed orchestration. SCA/GCM components can be equiped with a GCM membrane specially designed as a MAPE-compliant framework for flexible SOA applications. This means the SOA application can flexibly be recomposed and redeployed if needed, according to some SLAs, in an autonomic way [7].

GCM is also being used to program peer-to-peer applications. More specifically, it is used to build up a cloud-based system for storing and brokering semantically described (RDF) events, relying upon a structured CAN-based overlay architecture. Peers and associated proxies of this system named *Event cloud* are programmed so to fulfill the Event Level Agreements dictated by the services that are publishing or subscribing to events.

As a conclusion, GCM is becoming a mature technology for programming large-scale distributed and parallel applications on grids, clouds and any combination of them.

References

1. Aldinucci, M., Campa, S., Danelutto, M., Vanneschi, M., Kilpatrick, P., Dazzi, P., Laforenza, D., Tonellotto, N.: Behavioral skeletons in GCM: automatic management of Grid components. In: 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (2008)
2. Barros, T., Boulifa, R., Cansado, A., Henrio, L., Madelaine, E.: Behavioural Models for Distributed Fractal Components. Annals of Telecommunications 64(1) (2009)
3. Baude, F., Caromel, D., Dalmasso, C., Danelutto, M., Getov, V., Henrio, L., Pérez, C.: GCM: A Grid Extension to Fractal for Autonomous Distributed Components. Annals of Telecommunications 64(1), 5–24 (2009)
4. Baude, F., Henrio, L., Naoumenko, P.: Structural reconfiguration: an autonomic strategy for GCM components. In: 5th International Conference on Autonomic and Autonomous Systems (ICAS 2009), pp. 123–128. IEEE Xplore (2009)
5. Henrio, L., Khan, M.: Asynchronous Components with Futures: Semantics and Proofs in Isabelle/HOL. In: FESCA 2010. ENTCS (2010)
6. Mathias, E., Baude, F.: A Component-Based Middleware for Hybrid Grid/Cloud Computing Platforms. In: Concurrency and Computation: Practice and Experience (to appear, 2012)
7. Ruz, C., Baude, F., Sauvan, B.: Flexible Adaptation Loop for Component-based SOA applications. In: 7th International Conference on Autonomic and Autonomous Systems (ICAS 2011). IEEE Explorer (2011) Best paper awarded